64000

HP 64000 Logic Development System

Model 64341 Real-Time High Level Software Analyzer For 68000/68010

HEWLETT PACKARD

CERTIFICATION

Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

WARRANTY

This Hewlett-Packard system product is warranted against defects in materials and workmanship for a period of 90 days from date of installation. During the warranty period, HP will, at its option, either repair or replace products which prove to be defective.

Warranty service of this product will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all other cases, products must be returned to a service facility designated by HP.

For products returned to HP for warranty service, Buyer shall prepay shipping charges to HP and HP shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to HP from another country.

HP warrants that its software and firmware designated by HP for use with an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environment specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HP SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HP SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

ASSISTANCE

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales and Service Office.

KPR #: D200040972 **Product:** See product list on front cover. 02.01

Keywords: Z203

One-line description:
The analyzer does not work with #include statements in C or PASCAL

Problem:
The high level language analyzer does not work with code that contains #include statements whose files contain executable code. The source code of an included file is not available for display by the analyzer software as it is now written. Included files which contain only type declarations (or statements which are non-executable) pose no problems for the analyzer.

Solution:
If it is necessary to debug code which uses #include statements of files which contain executable code, the user (for the purposes of debugging those executable statements) may merge in the file and eliminate the include statement while he is debugging that portion of the code. After debugging that portion of the code, he can replace the code with the original include statement.

SAFETY SUMMARY

The following general safety precautions must be observed during all phases of operation, service, and repair of this instrument. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the instrument. Hewlett-Packard Company assumes no liability for the customer's failure to comply with these requirements.

GROUND THE INSTRUMENT.

To minimize shock hazard, the instrument chassis and cabinet must be connected to an electrical ground. The instrument is equipped with a three-conductor ac power cable. The power cable must either be plugged into an approved three-contact electrical outlet or used with a three-contact to two-contact adapter with the grounding wire (green) firmly connected to an electrical ground (safety ground) at the power outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE.

Do not operate the instrument in the presence of flammable gases or fumes. Operation of any electrical instrument in such an environment constitutes a definite safety hazard.

KEEP AWAY FROM LIVE CIRCUITS.

Operating personnel must not remove instrument covers. Component replacement and internal adjustments must be made by qualified maintenance personnel. Do not replace components with the power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

DO NOT SERVICE OR ADJUST ALONE.

Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

DO NOT SUBSTITUTE PARTS OR MODIFY INSTRUMENT.

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification of the instrument. Return the instrument to a Hewlett-Packard Sales and Service Office for service and repair to ensure that safety features are maintained.

DANGEROUS PROCEDURE WARNINGS.

Warnings, such as the example below, precede potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

WARNING

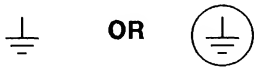
Dangerous voltages, capable of causing death, are present in this instrument. Use extreme caution when handling, testing, and adjusting.

SAFETY SYMBOLS

General Definitions of Safety Symbols Used on Equipment or in Manuals.

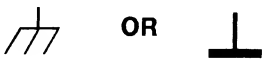
Instruction manual symbol: the product is marked with this symbol when it is necessary for the user to refer to the instruction manual in order to protect against damage to the instrument.

Indicates dangerous voltage (terminals fed from the interior by voltage exceeding 1000 volts must be so marked).



Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.

Low-noise or noiseless, clean ground (earth) terminal. Used for a signal common, as well as providing protection against electrical shock in case of a fault. A terminal marked with this symbol must be connected to ground in the manner described in the installation (operating) manual, and before operating the equipment.



Frame or chassis terminal. A connection to the frame (chassis) of the equipment which normally includes all exposed metal structures.

Alternating current (power line).

Direct current (power line).

Alternating or direct current (power line).

The WARNING sign denotes a hazard. It calls attention to a procedure, practice, condition or the like, which, if not correctly performed, could result in injury or death to personnel.

The CAUTION sign denotes a hazard. It calls attention to an operating procedure, practice, condition or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product.

NOTE:

The NOTE sign denotes important information. It calls attention to procedure, practice, condition or the like, which is essential to highlight.

HEWLETT PACKARD

OPERATING MANUAL

MODEL 64341
REAL-TIME HIGH LEVEL SOFTWARE ANALYZER FOR 68000/68010

Manual Part No. 64341-90903
E0985

PRINTED: **September 1985**

PRINTING HISTORY

Each new edition of this manual incorporates all material updated since the previous edition. Manual change sheets are issued between editions, allowing you to correct or insert information in the current edition.

The print date changes only when each new edition is published. Minor corrections or additions may be made as the manual is reprinted between editions. Vertical bars in a page margin indicates the location of reprint corrections.

First Printing ..... Feburary, 1985 (64341-90901)
Second Edition .....September, 1985 (64341-90903)

SOFTWARE VERSION NUMBER

Your HP 64000 software is identified with a version number in the form YY.XX. The version number is printed on a label attached to the software media or media envelope. This manual applies to the following:

Model HP 64341BA	Version 2.XX
Model HP 64341DA	Version 2.XX
Model HP 64341GA	Version 1.XX
Model HP 64341IA	Version 1.XX

Within the software version number, the digit to the left of the decimal point indicates the product feature set. This manual supports all software versions identified with this same digit.

The digits to the right of the decimal point indicate feature subsets. These feature subsets normally have no affect on the manual. However, if you subscribe to the "Software Material Subscription" (SMS), these subset items are covered in the "Software Response Bulletin" (SRB).

SOFTWARE MATERIALS SUBSCRIPTION

Hewlett-Packard offers a Software Materials Subscription (SMS) to provide you with the most timely and comprehensive information concerning your HP 64000 Logic Development System. This service can maximize the productivity of your HP system by ensuring that you have the latest product enhancements, software revisions, and software reference manuals.

For a more detailed description of the SMS, refer to chapter 1.

DUPLICATING SOFTWARE

Before using the flexible disc(s) provided with this product, make a work copy. Retain the original disc(s) as the master copy and use the work copy for daily use. The procedure for duplicating the master flexible disc(s) is included in chapter 2 of this manual.

Specific rights to use one copy of the software product(s) are granted for use on a single, standalone development station or a cluster of development stations which boot from a single mass storage device.

Should your master copy become lost or damaged, replacement discs are available through your Hewlett-Packard sales and service office.

TABLE OF CONTENTS

Chapter 1. GENERAL INFORMATION

Chapter 2. INSTALLING THE SOFTWARE ANALYZER

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont'd)

LIST OF ILLUSTRATIONS (Cont'd)

LIST OF TABLES

NOTES

USING THIS MANUAL

The contents of this manual are summarized below to aid you in locating information.

Chapter 1, General Information, provides an overview of the software analyzer.

Chapter 2, Installation, describes the system components required to run the analyzer package and the procedures for installing those components.

Chapter 3, Getting Started, takes you through the entire measurement process step-by-step and gives guidelines for writing code to achieve the best results from your software analyzer.

Chapter 4, Building Database Files, describes how to build database files, how to verify database files are correct, and the implications of using compiler directives with the analyzer.

Chapter 5, Defining Measurement Parameters, describes how to define several analyzer global parameters affecting measurements.

Chapter 6, Qualifying Measurements, describes the use of measurement enable and disable terms to qualify measurements.

Chapter 7, Controlling the Emulator, gives information on loading and running programs with the emulation system from within the software analyzer.

Chapter 8, Making Trace Measurements, gives descriptions of each of the trace measurements.

Chapter 9, Making Count and Time Measurements, describes in detail the Count Statements and Time Modules measurements.

Chapter 10, Using Interactive Commands For Program Debugging, describes how to use hardware breaks and the display and modify commands to interact with the user program.

Chapter 11, Making Intermodule Bus Measurements, gives detailed information, including examples, on making intermodule bus measurements.

Chapter 12, Selecting and Formatting the Measurement Display, describes the conventions and features of the measurement display, and the commands used to format the measurement display.

Chapter 13, Configuring the Analyzer, describes how to both manually and automatically configure the analyzer for measurements.

Chapter 14, Using Support Commands, describes system software conventions and utility commands available within the software analyzer.

Chapter 15, Symbols and Data Types, provides information regarding the symbol storage classes and data types that the software analyzer recognizes.

Chapter 16, Operational Theory, provides a description of how measurements are made and system resources used.

Appendices A through E provide operating syntax diagrams, status and error messages, stack and memory organization, softkey prompts, a softkey glossary, and solutions to measurement problems.

An index is provided for quick reference to specific items.

Chapter 1

GENERAL INFORMATION

OVERVIEW

This chapter answers the following questions:

- Which products does this manual apply to?
- What is a real-time high level software analyzer?
- What does the software analyzer allow you to do?
- How can you use the software analyzer in heirarchical Measurements?
- What are the conventions used in examples in this manual?
- What the software materials subsription can do for you?

SAFETY CONSIDERATIONS

This product is a Safety Class 1 instrument (provided with a protective earth terminal) and meets safety standards IEC 348. Review the instrument and this manual for safety markings and instructions before operating the instrument.

MANUAL APPLICABILITY

This manual applies to the following Real-Time High Level Analyzer products:

HP 64341BA for use with HP 64242S Emulation Systems (for 68000 processors)

HP 64341DA for use with HP 64249S Emulation Systems (for 68010 processors)

HP 64341GA for use with HP 64243AA/AB Emulation Systems (for 68000 processors)

HP 64341IA for use with HP 64245AA/AB Emulation Systems (for 68010 processors)

These real-time high level analyzers require that your have the HP 64340A Real-Time High Level Analyzer hardware, the HP 64000 emulation system for your processor, and an HP 64815 Pascal and/or HP 64819 C Cross Compiler. See chapter 2 for a detailed breakout of software an hardware compatibility requirements.

NOTE

Unless otherwise specified, explanations and examples in this manual apply to all real-time high level software analyzers listed in the preceding paragraphs. Most examples in this manual were generated using the HP 64341BA Real-Time High Level Software Analyzer for 68000 processors.

WHAT IS A REAL-TIME HIGH LEVEL SOFTWARE ANALYZER

The real-time high level software analyzer (hereafter referred to as "software analyzer") is a hardware and software system with the ability to perform measurements in real-time on your software executing in an HP 64000 emulation environment without interrupting execution of your code. The special hardware and software of the HP 64340A Analyzer allows it to analyze emulation bus signals in real time; routines can be timed, interrupt modules can be studied, and other analyzers can be triggered while the emulator is running at full speed.

The software analyzer is a three board emulation bus analyzer, which uses the emulator subsystem and the database file created by the HP Pascal and C compilers and linkers. The analyzer performs measurements on Pascal or C programs running in the emulator in real time, without halting the emulator, and displays the data in the same high level language constructs in which the software designer wrote the code.

In real-time mode, the software analyzer is fully transparent to the system under test and meet all criteria for real-time analysis. The processor is not halted, program execution is not stopped, and additional code and traps are not added to the target software. By running in non-real time mode (hardware breaks to the emulation monitor allowed), the software analyzer can access additional information by breaking the emulator and examining registers or memory locations.

Hardware breaks are used to halt the emulator when necessary, either when specified by the measurement or when running non-real time mode. The analyzer captures data in much the same way as other real time analyzers. Data patterns or addresses on the emulation bus are recognized by specialized high speed comparators, then the required information is stored in acquisition memory. However, because of the complexity of the new measurements performed, a great deal of specialized hardware is necessary.

HP 64340A Hardware Description

The HP 64340A hardware consists of three boards: a CPU board, an Acquisition board, and a Control board. The CPU board contains a 68000 microprocessor used to control the analyzer, perform the measurements, and process the stored data. A custom HP integrated circuit chip provides high speed complex recognition capabilities. The CPU board has 1/2 megabytes of dedicated RAM for the 68000 processor to work with. The analyzer has a data storage capacity of 96 bits X 4K. The 96 channels consist of 24 emulation address bits, 16 data bits, 8 status bits, 24 program counter bits, 20 bits for a time/state tag, and 4 flag bits. This information is stored during measurement execution. When the memory is full (or measurement complete) the 68000 processor postprocesses the data to remove unwanted information and to reference the source code.

Figure 1-1 is a functional block diagram of the software analyzer hardware. The on-board 68000 processor initializes and sets up the hardware for each measurement, communicates with the HP

64000 host processor, and is the computing engine of the analyzer. The high speed general purpose state machines and function generators are the controllers of the measurement while it is being executed. For example, these components determine if sequence conditions are met and load the dynamic recognition resources, as well as perform numerous other tasks.

The static and dynamic recognition resources are custom high speed comparators, the only difference being that the dynamic recognition patterns must be loaded during execution. This is necessary because the locations of dynamic variables are not known until execution.

Each stored state in the 96 channel data storage memory contains address and data information from the emulation bus, time/state tags, flags from the state machines, and the computed PC. This "last PC" is computed by using a special "opcode fetch" equate resource. At the start of an instruction, this equate goes true, and the current PC is saved until the start of the next instruction.

The final block on the diagram is labeled count statements. This special hardware consists of 256 counters 4K deep, and is devoted to this one measurement.

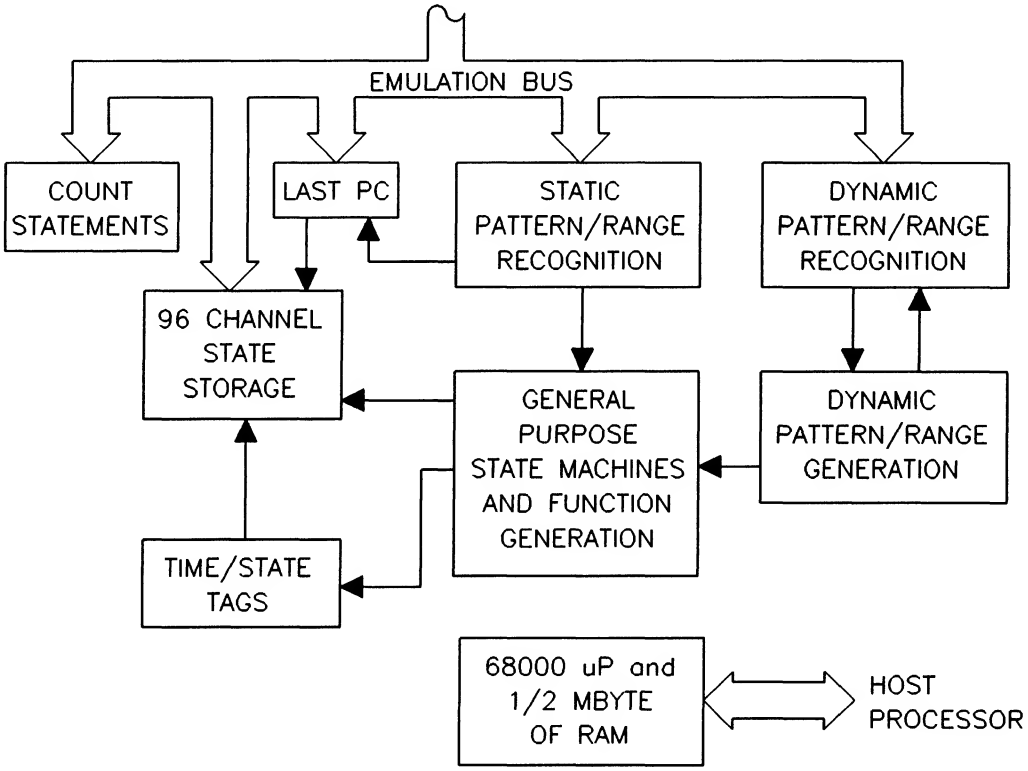


Figure 1-1. HP 64340A Hardware Functional Block Diagram

HP 64341 Software Description

A functional block diagram of the HP64341 Software is shown in figure 1-2. The two main functions of the software are to communicate with the HP 64000 host processor and to control the measurements. The modules associated with the interface control the passing of messages back and forth between the emulator and the software analyzer, setting up the IMB, and providing an application monitor. The core of the application software is embodied in the block labeled

measurement controllers. This software sets up the analyzer to capture the correct data, and then analyzes the captured data.

Since information flowing over the emulation bus is in low level code, much processing must be done both to set up the measurement and to display the final result in high level symbols and code. The on-board 68000 microprocessor performs this processing by accessing the database and symbol files created at compile and link times. When the measurement is specified, in terms of high level variables, procedure names, or line numbers, the processor must translate these into the low level constructs to set up the hardware correctly. During the execution of the measurement, the 68000 processor initiates the measurement and oversees the entire process. It is the interface to the host processor and, through the host processor, the interface with the emulator. Note: the emulator doesn't participate actively in the acquisition of data.

When a measurement is complete, the 68000 processor postprocesses the data in acquisition memory. Much of the data acquired may be irrelevant; prefetched instructions can be filtered out, and much of the saved stack information may not be useful for a particular measurement. Furthermore, the on-board processor displays the measurement in the original high level language constructs. This requires referencing symbol tables and considerable processing on the part of the on-board 68000 processor.

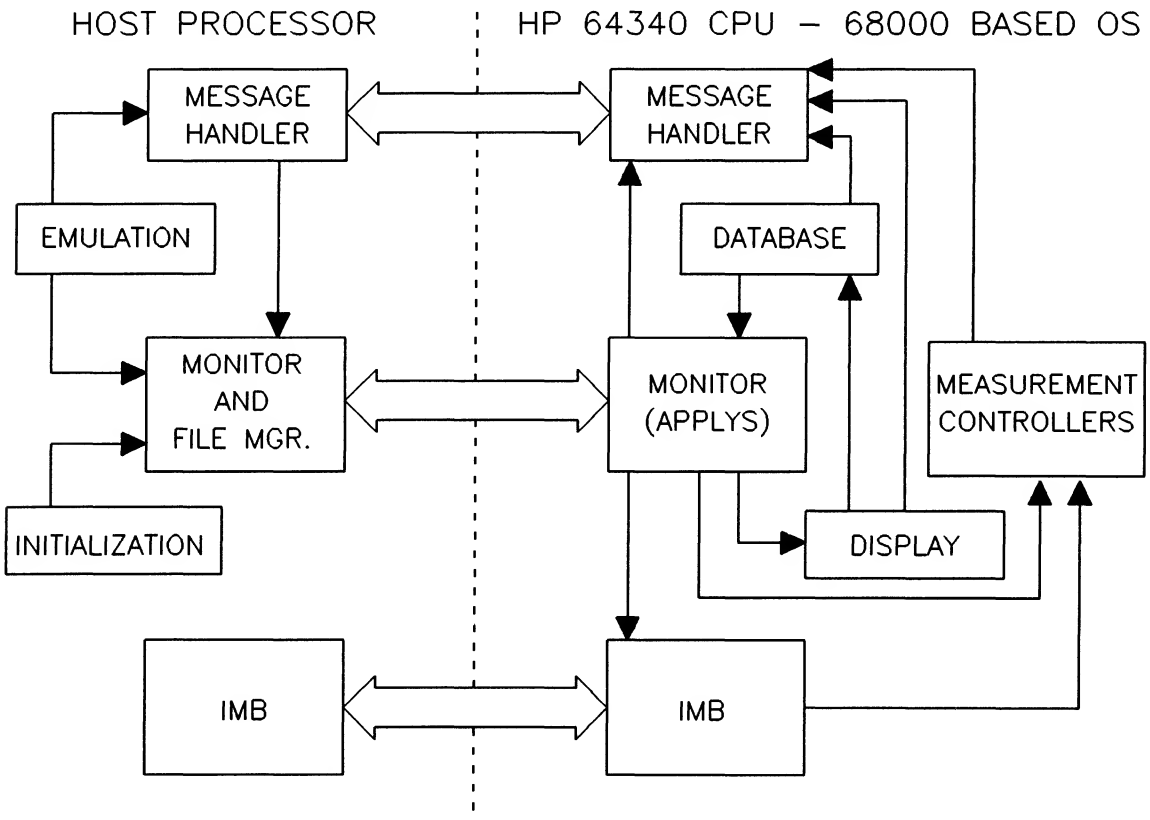


Figure 1-2. HP 64341 Software Functional Block Diagram

WHAT THE SOFTWARE ANALYZER ALLOWS YOU TO DO

Trace Measurements

The software analyzer has four trace measurement modes. Trace modules and trace data flow are global measurements, giving you an overview of both program and data flow at the module level. Trace statements and trace variables are local measurements which give the precise order of statement execution or values of specific variables every time they are accessed.

TRACE MODULES. The trace modules measurements tracks program flow by capturing the entry and exit points to the specified modules. This is useful in many situations: often modules are written by different programmers and may even be in different high level languages. Tracing module flow when the modules are first integrated shows what order they are called in and indicates possible locations of problems.

Either specifically named modules or all the modules in a file can be traced. Modules can be in up to four non-adjacent or ten adjacent files. The analyzer can trace recursive calls indefinitely and can trace both Pascal and C modules in the same measurement.

The trace modules measurement can run in both real-time and non-real time mode. In real-time mode, all modules files can be traced to see program flow, including interrupt routines. Accurate time tags are displayed which indicate the time spent in each module. Thus, you can quickly see the order in which the modules are executed, when recursion occurred, how often an interrupt routine was called, and how much time was spent in each module.

No information is lost running non-real time, however the emulator is halted approximately every 100mS and the time tags include this time. The benefits to running in non-real-time is that windowing is allowed and more resources can be used when enabling the measurement using sequenced terms (explained in the Measurement Control paragraphs later in this chapter).

This measurement is useful in locating problems to a general area of your program. If any module execution deviates from expectations, another measurement can be made to localize the problem.

Figure 1-3 is a trace modules measurement display. The display shows entry and exit points of a module, shows nesting and recursion, gives an accurate time/state count, and displays the source line calling the module upon entry. Each field can be formatted to meet your requirements. Time can be displayed in relative or absolute mode. Selection of time or state count must be specified before the execution of a measurement.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Symbol                    Stat   Time-rel Source

PROC1                     entry  993.4 uS  230 PROC1 (COUNT,COUNT+2);
 RECURSIVE_PROC           entry  313.3 uS  145 RECURSIVE_PROC (FPARM1, FPARM1,
  RECURSIVE_PROC          entry  623.0 uS  120 RECURSIVE_PROC (RP1,RP2,RP3,RP4,
   RECURSIVE_PROC         entry  615.0 uS  120 RECURSIVE_PROC (RP1,RP2,RP3,RP4,
    RECURSIVE_PROC        entry  617.3 uS  120 RECURSIVE_PROC (RP1,RP2,RP3,RP4,
    RECURSIVE_PROC        exit   6.557 mS
   RECURSIVE_PROC         exit   6.020 mS
  RECURSIVE_PROC          exit   5.982 mS
 RECURSIVE_PROC           exit   5.932 mS
PROC1                     exit   589.2 uS
PROC2                     entry   12.3 uS  231 PROC2 (COUNT+2);
 NESTED_PROC              entry   11.2 uS  167 NESTED_PROC (A);
 NESTED_PROC              exit     5.4 uS
PROC2                     exit     5.9 uS

STATUS: Awaiting Command                                          36   16:45

   run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 1-3. Trace Modules Measurement Display

TRACE DATA FLOW. The trace data flow measurement traces the values of data at the entry and exit points of a procedure. Both static and dynamic variables can be traced. Data pointed to by up to seven levels of pointers can be accessed and displayed in this measurement. Unlimited recursion can also be traced. Up to three different modules can be traced in one measurement, with up to 10 symbols specified.

Local variables and variables passed by value cannot be displayed at the exit point of the procedure. At this point, they are undefined (they have been popped off the stack). Since the traced data is not accessible on the emulation bus at entry and exit points of a module, this measurement must be run in non-real-time.

This measurement allows you to view data at entry and exit of a procedure, showing whether it was modified within the module. The values of variables can be seen at each level of a recursive procedure. This is very useful if a procedure is stuck in infinite recursion. The variable which should cause an exit condition can be traced and the bug quickly found.

Figure 1-4 is a trace data flow measurement display. The specified variable values are displayed at entry or exit points of modules (or both), and the source code line number that called the procedure is displayed on entry.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Symbol                Value          Stat  Source
PROC4                                entry  183 PROC4(COUNT+2);
  X                          -1
PROC4                                exit
  X                          -1
PROC10                               entry  201 PROC10(X,X,Y,Y);
  XV                         10
  XN                         10
  YV                  00000300CH
  YN                  00000300CH
  A[RED]                    RED
PROC10                               exit
  XN                         11
  YN                  00000300CH
  A[RED]                    RED
PROC4                                entry  183 PROC4(COUNT+2);

STATUS: Awaiting Command                                        20    16:12


   run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 1-4. Trace Data Flow Measurement Display

TRACE STATEMENTS. The trace statements measurement traces statement flow within a single module. The statements are displayed in the order of their execution and variable values are displayed. The measurement can run in both real-time and non-real-time. The statement range can be defined as the entire procedure or a line range within a procedure. There is also a "don't care" specification, which traces everything flowing over the emulation bus. When using the "don't care" specification, no variable values are displayed. This specification should only be used in real-time required mode. Otherwise the acquisition memory fills up with useless monitor information.

Only the values of static variables are displayed when running in real-time required mode. Dynamic variables can be traced in non-real-time. Figure 1-5 is a trace statements measurement display. Again, time tags are displayed which give an accurate execution time for each high level statement.

This measurement is useful when a problem has been isolated down to a module. The display gives a step-by-step view of the execution order of the high level statements much like a state display does with low level code. The debugging process can be greatly sped up, as all the relevant information concerning the execution of a module is displayed. This is a highly effective way to observe the interaction between program and data flow.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Source                                    Symbol      Value       Stat   Time-rel

Break for new stack information
  82 PTR^.I := PTR^.I-1;                  PTR          000003028H read      0.0 uS
  83 Y:=1;                                                                 4.3 uS
  84 D:=D-1; (*Scoped variable*)          Y          1.00000E0 write     481.4 uS
  85 P2:=SNN; (*STATIC  CALLBYNAME  *     D                  4 read      4.2 uS
                                          D                  3 write     2.4 uS
                                          SNN                2 read      3.6 uS
  86 P2:=SNN; (*STATIC  CALLBYNAME  *     P2                 2 write     1.3 uS
                                          SNN                2 read      5.0 uS
  87 P2:=SNV; (*STATIC  CALLBYNAME  *     P2                 2 write     1.2 uS
                                          SNV                2 read      3.4 uS
  88 P2:=SVN; (*STATIC  CALLBYVALU  *     P2                 2 write     1.0 uS
                                          SVN                4 read      5.1 uS
  89 P2:=SVV; (*STATIC  CALLBYVALU  *     P2                 4 write     1.0 uS

STATUS: Awaiting Command                                      36    16:12

   run     setup   db check   display   modify    show     execute   ---ETC---
```

Figure 1-5. Trace Statements Measurement Display

TRACE VARIABLES. The trace variables measurement allows you to trace all accesses to specified variables during program execution. The measurement can run both in real-time and non-real-time. The measurement functions for both modes are identical. No objects of pointers can be referenced, and only the values in the outer layer recursion are displayed.

Figure 1-6 shows a trace variables measurement display. Source line numbers are displayed, making this a very useful localized debugging tool. A variable which is seen to have an incorrect value can then be traced, and all the reads and writes to it displayed. It is then a simple matter to determine where the program went astray.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

symbol                Value        stat  source
pred_result.enumera*          red write   17 pred_result.enumerated = red;
pred_result.arr[0]              0 read   158 check = check + pred_result.arr[0]
pred_result.u8                 50 read    28 pred_result.u8 = 0;
pred_result.u8                  0 write   28 pred_result.u8 = 0;
pred_result.arr[0]              0 read   162 check = check + pred_result.arr[0]
pred_result.s16             -7166 write   39 pred_result.s16 = -1BFEH;
pred_result.enumera*        green write  166 pred_result.enumerated = green;
pred_result.arr[0]              0 read   167 check = check + pred_result.arr[0]
pred_result.arr[1]              0 read   169 check = check + pred_result.arr[1]
pred_result.enumera*         blue write  171 pred_result.enumerated = blue;
pred_result.ch                "A" write  174 pred_result.ch = 'A';
pred_result.arr[0]              0 read   175 check = check + pred_result.arr[0]
pred_result.arr[1]              0 read   177 check = check + pred_result.arr[1]
pred_result.ch                "a" write  179 pred_result.ch = 'a';
pred_result.s16              3700 write   85 pred_result.s16 = 3700;

STATUS: Awaiting Command                                          30    16:12


   run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 1-6. Trace Variables Measurement Display

Count/Time Measurements

TIME MODULES. Figure 1-7 shows a time modules measurement display. The time modules measurement can time up to four modules, and displays the minimum, maximum, and mean time spent in each module. The time includes all time between entry of the specified module and exit from that module, including time spent in subroutines and servicing interrupts. The software analyzer can time recursive modules, up to 256 levels deep. The measurement can be run in both real-time and non-real-time. In non-real-time the emulator is halted in the order of microseconds every 100 milliseconds. Therefore, if the measurement is used for estimates, this will not affect the results substantially.

This measurement is useful in a variety of cases. Modules can be tested to see if they are executing within specified times. Inefficient modules can be found and then optimized. Also, the effect of interrupts on modules can be studied. The display also shows the number of times the module was timed, giving an indication of the statistical accuracy of the measurement.

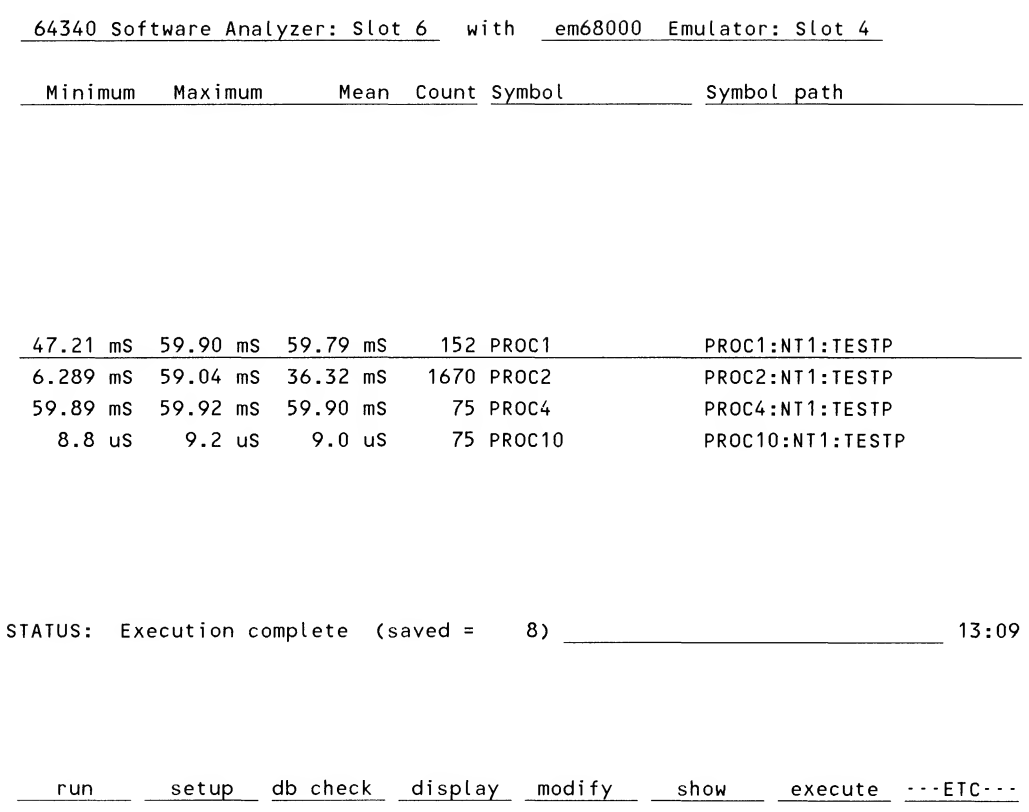


Figure 1-7. Time Modules Measurement Display

COUNT STATEMENTS. A count statements measurement display is shown in figure 1-8. The count statements measurement counts the number of times each statement in a specified module or line range is executed. Up to 255 statements can be counted but they all must be in one module.

The main application of this measurement is in the area of software coverage testing. In the testing phase of software development, it is often difficult to know whether all of the software has been exercised. For example, a certain branch may never be taken or parts of a case statement may never be executed. Count statements is a simple method to verify this coverage testing. If a statement is never executed, either another test can be run to exercise it, or it can be removed if it will never be executed. Also, the count statements is an easy way to verify the operation of loop counters, allowing you to verify that the statements within the loop were executed the specified number of times.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Count-abs Source

      682   94 IF COUNT = 10
       62   95 THEN  COUNT:=0
        0   96 ELSE
        0   97 BEGIN
      620   98 COUNT := COUNT+1;
      620   99 COLOR_SET := COLOR_SET + [ WHITE, GREEN ];


STATUS: Awaiting command                                    0 __ 13:09

   run    setup   db check   display   modify    show    execute  ---ETC---
```

Figure 1-8. Count Statements Measurement Display

Break Measurement

The break measurement allows up to nine user definable hardware breakpoints to be executed. These are hardware breakpoints and can be set up anywhere in the user code, even in ROM. This is a measurement; no other measurement can be set up concurrently. A hardware break can be set up at the end of other measurements. That is discussed in the emulation control paragraphs in this chapter.

The break measurement display shows the last line of executed code before the break was executed. A variable can then be displayed (useful for tracing pointer objects) or modified and program execution then started again from the breakpoint. If the emulator is set up correctly, these breaks can force a jump into any part of the user code instead of the monitor. This is useful forcing interrupt routines to occur at specific times in the execution of the program.

Emulation Control

Many functions of the emulator can be initiated or controlled by the software analyzer. All of these functions can be specified after the measurement is set up, and a few can be specified beforehand.

LOAD. The load softkey is used to load the absolute file into emulation/user memory. It functions the same as the load softkey in the emulation system.

RUN. The run command is similar to the emulation system run command. It starts the emulator from a specified address, a symbolic location, the transfer address, or if nothing is specified, the next PC. The program can be specified to run at execution, which causes the program to run when the execute key is pressed and the measurement is initiated.

BREAK. The break key issues an immediate hardware break to the emulator.

RESET. The reset key immediately resets the emulator.

BREAK ON MEASUREMENT COMPLETE. When "break on measurement complete" is specified before measurement initiation, the analyzer breaks the emulator at the end of the measurement. The emulator can then be started at the next PC and no data will be lost. A measurement is complete when either the acquisition memory is full, or the disable condition has occurred.

Software Control

Various features of the software analyzer allow you to examine and verify source code, and modify program variables without exiting the software analyzer. These features save time. The ability to modify high level variables helps greatly in the debugging process.

SHOW SOURCE. This command allows you to display any HP 64000 source file while using the software analyzer. If no arguments are specified in the command, the default path source file is displayed. Editing features are not available, but the user can scroll up and down through the file or position the screen using a line number.

DATABASE CHECK. With this command, you can verify the software version that the analyzer is using. All the absolute files being used are check against the comp_db files to make sure that the files have not been recompiled but not reloaded.

DISPLAY VARIABLE. This command allows you to display the value of any variable. Pointer indirection can be traced through seven levels to reach the data object. In order to execute this command, the emulator must be running in the monitor program and the variable must be scoped.

MODIFY VARIABLE. You can modify any high level variable that can be displayed, i.e., that is currently scoped. This allows you to change a variable without having to recompile and relink the code. The emulator must be running in the monitor to perform this command.

Measurement Control

Measurement control features allow you to control when a measurement is executed, how much data is collected, and what part of the user code that is executing should be measured.

STARTING AND STOPPING MEASUREMENTS. Standard analysis softkeys are used to start and stop measurements. *execute* starts the measurement. The analyzer either searches for an enable term or begins searching for and collecting measurement data. If IMB triggering is used, measurement execution is tied to all other execute softkeys in the measurement system , enabling synchronous measurement.

While the measurement is executing, two softkeys are present, *wait* and *halt*. the wait command is used in command files so that a measurement can be executed from a command file, and then more commands can be issued. The command file can wait for a specified number of seconds, for any keystroke, or for the measurement to be completed. The halt command halts the execution of the measurement before all data has been acquired. After all data has been acquired, the halt softkey remains while acquisition memory is unloaded and the data is being postprocessed. The halt softkey does not generate a measurement complete flag.

CONTROLLING THE MEASUREMENT WINDOW. Measurement enable and disable terms control the window of user code viewed by the analyzer. The measurement enable term allows up to six levels of sequencing using symbols. Each level can also have OR'ed terms. In real-time mode, the disable term can also have up to six levels of sequencing. Windowing to reenable the measurement is not allowed. In non-real time, sequencial disable terms are not allowed. A single disable term is allowed, and windowing can be specified to restart the measurement when the next enable term is found.

MODIFYING MEASUREMENT SETUPS AND DISPLAYED DATA. The measurement setup and the displayed data can be modified quickly and easily. The measurement specification or the enable or disable specification can be modified without re-entering all the terms. This is useful with complex measurements. The displayed data can also be modified. The width of the fields can be changed if all the data can't be displayed, and the values of variables can be displayed in different bases or in ASCII characters.

IMB MEASUREMENTS. Analyzer measurements can control (or be controlled by) the trigger enable line on the intermodule bus (IMB). The IMB can be driven by the occurrence of either a measurement enable or disable term. A received signal can either cause an enable or disable.

MAKING HIERARCHICAL MEASUREMENTS

Applying software analysis measurements in a top down sequence (hierarchical) is very useful when there is little initial information about the cause of a software failure. At a coarse or global level, the Trace Modules measurement can verify that procedures and functions are executed in the proper sequence and at the appropriate nesting levels. If an incorrect sequence or nesting level is found, the Trace Statements measurement can determine the precise location of a software fault. If the modules occur in the correct sequence and level, the Trace Data Flow measurement can point out incorrect parameter values and global variables passed to and from selected modules.

Assuming module execution sequences and parameters values are correct, the Trace Statements measurement displays program flow in more detail. The Trace Statements measurement, showing executed source lines and values of referenced global and local variables, allows you to distinguish between errors caused by programming flaws and those caused by unexpected variable values. A

Trace Variables measurement can then be applied to isolate the cause of improper variable assignments.

Once the software is executing properly, the Count Statements and Time Modules measurements allow coverage testing and performance analysis of the software modules. The Count Statements measurement shows the number of times a source statement or range of source statements are executed. The Time Modules measurement measures the real-time execution speeds of up to four modules, pinpointing bottlenecks that may require recoding.

UNDERSTANDING THE EXAMPLES USED IN THIS MANUAL

The examples provided throughout this manual use the following structure:

PRESS (or press) *edit* MODULE (RETURN).

PRESS or press-- means you should enter a command by selecting the softkeys and/or typing in any file names or other variables which are not provided in the softkey selections.

edit -- softkeys will appear in italics. Usually you will not be prompted to use the ---*ETC*--- softkey to search for the appropriate softkey template.

MODULE -- this is the name of a file which you must type in. Softkeys are not provided for this type of selection since it is variable. However, a softkey prompt such as, <*FILE*> will appear as a softkey selection.

(RETURN) -- this indicates that the RETURN key, located on the keyboard, should be pressed.

SOFTWARE MATERIALS SUBSCRIPTION

Hewlett-Packard offers a Software Materials Subscription (SMS) to provide you with the most timely and comprehensive information concerning your HP 64000 Logic Development System. This service can maximize the productivity of your HP system by ensuring that you have the latest product enhancements, software revisions, and software reference manuals.

Consult with your local HP Field Representative for a complete list of available software update products (HP 64XXXAU), one-time product updates (HP 64XXXAX), and current prices.

By purchasing SMS, you will obtain the following:

Software Updates
Reference Manual Updates
Software Problem Reporting
Software Release Bulletins
Software Status Bulletins
General User Information

Software Updates

Software Updates may address specific anomalies in HP software or enhance the capability of the HP software in your system.

Reference Manual Updates

Reference manual updates assure that you always have the most recent documentation on a timely basis, and are aware of how to use any new features on the latest software releases.

Software Problem Reporting

Software problem reporting is provided so that you may inform HP of a discrepancy or problem found in the HP 64000 software or documentation.

Software Release Bulletins

Software Release Bulletins document all fixes and enhancements that are incorporated in the latest release of the HP 64000.

Software Status Bulletins

Software status bulletins contain timely information on the reported operational status of HP software and documentation. These bulletins also provide temporary corrections or ways to work around anomalies in HP software which have been located by HP personnel or HP 64000 users. You may reference these bulletins to see if a solution is already documented.

General User Information

General user information is documentation that contains operational tips, programming techniques, application notes, latest listings of software products and reference manuals, and other items of general interest to HP 64000 users.

NOTES

Chapter 2

INSTALLING THE SOFTWARE ANALYZER

OVERVIEW

This chapter provides the following information:

- A complete list of software analyzer hardware and software
- How to install the software analyzer hardware
- How to install the software analyzer software
- How to make duplicate copies of floppy disc software
- How to perform a software analyzer operation verification

INTRODUCTION

The software analyzer software is shipped on three floppy discs. In addition to the software, you must have the Model 64340A Software Analyzer hardware,the HP 64000 hosted Pascal and/or C cross compiler for 68000/68010 processors, and the HP 64000 Emulation System for your processor. If your Model 64100A development station has a serial number prefix lower than 2309A, you also need a Model 64032A Memory Expansion Module in order to compile Pascal or C programs on your development station.

HARDWARE AND SOFTWARE REQUIRED FOR HIGH LEVEL SOFTWARE ANALYSIS

The following HP 64000 software and hardware products are required for real-time high level software analysis.

Software Analyzer Software

HP 64341BA Software Analyzer Software for use with HP 64242S Emulation Systems (for 68000 processors) consisting of five (5) software modules contained on three (3) flexible discs:

ANLY_341_68000_1
ANLY_341_68000_2
DB_68000
GEN_DB_PASCAL
GEN_DB_C

HP 64341DA Software Analyzer Software for use with HP 64249S Emulation Systems (for 68010 processors) consisting of five (5) software modules contained on three (3) flexible discs:

ANLY_341_68010_1
ANLY_341_68010_2
DB_68010
GEN_DB_PASCAL
GEN_DB_C

HP 64341GA Software Analyzer Software for use with HP 64243AA/AB Emulation Systems (for 68000 processors) consisting of five (5) software modules contained on three (3) flexible discs:

ANLY_341_68000D_1
ANLY_341_68000D_2
DB_68000
GEN_DB_PASCAL
GEN_DB_C

HP 64341IA Software Analyzer Software for use with HP 64245S Emulation Systems (for 68010 processors) consisting of five (5) software modules contained on three (3) flexible discs:

ANLY_341_68010D_1
ANLY_341_68010D_2
DB_68010
GEN_DB_PASCAL
GEN_DB_C

Software Analyzer Hardware

64340A Software Analyzer hardware consisting of:

one (1) CPU/Memory board

one (1) Acquisition board

one (1) Control board

three (3) Interconnection cables

Additional HP 64000 System Components Required

In addition to the software analyzer components listed above, you will need the following HP 64000 system components:

Model 64242S 68000 Emulation System (if using 64341BA analyzer)

Model 64249S 68010 Emulation System (if using 64341DA analyzer)

Model 64243AA/AB 68000 Emulation System (if using 64341GA analyzer)

Model 64245AA/AB 68010 Emulation System (if using 64341IA analyzer)

Model 64155A Wide Address Memory Control Board

Model 64161A or 64162A or 64163A Static RAM Board(s)

Model 64815AF or 64815S Pascal Cross Compiler

Model 64819AF or 64819S C Cross Compiler

INSTALLING ANALYZER HARDWARE

WARNING

Any servicing, adjustment, maintenance, or repair of this product must be performed only by service-trained personnel who are aware of the hazards involved.

Configuring boards in the station

Table 2-1 shows the preferred configuration of a 64100A development station containing a 64340A High Level Software Analyzer with various system options installed. The 64340A boards must be installed in the system from low to high numbered card slots as follows: The CPU board in the lower numbered card slot, the Control board in the next slot, and the Acquisition board in the higher numbered card slot. The lowest numbered card slot is closest to the front of the development station. See figure 2-3.

Installing The Analyzer In A 64100 Development Station

CAUTION

The three 64340A circuit boards are susceptible to damage from static discharge. To avoid damage, handle boards from the sides ONLY. Never touch the bottom of a circuit board.

1. Turn off power to the 64100A station.

2. Remove the four screws securing the cardcage cover to the top of the 64100A station. See figure 2-1.

3. Connect the three interconnection cables (W1) to J2, J3, and J4 of the Acquisition board (A3). See figure 2-2.

4. Install the CPU board first. Note the position of the Acquisition board to relative to other options installed in the station. See table 2-1.

5. Install the other two software analyzer boards in the station cardcage. The position for the three board 64340A card set in the cardcage should always be the CPU board in the lower numbered card slot, the Control board next, and the Acquisition board in the higher numbered slot. See figure 2-3.

6. Connect the three interconnection cables (W1) that were connected the acquisition in step 3 to the top of the 64340A Control and CPU boards. See figure 2-3.

Installing The Emulation System

The emulation boards (emulation control board, memory control board, memory board, etc.) should be placed in lower numbered card slots in the cardcage than the software analyzer). See Table 2-1 and the emulator operating manual for detailed installation procedures for the emulation system.

Installing Other Analysis Boards.

Other emulation/analysis boards should be placed in higher numbered card slots than the 64340A Software Analyzer. Refer to table 2-1 and the installation chapter of the specific emulator/analyzer manual for detailed installation procedures.

NOTE

If you are using a model 64310A Software Performance Analyzer with the real-time high level software analyzer, the 64310A software module **SW_PERF_ANALYZER** must have a software revision number of 1.11 or greater. The software analyzer is incompatible with earlier versions of the 64310A Software Performance Analyzer.

Figure 2-1. Cardcage Cover Removal

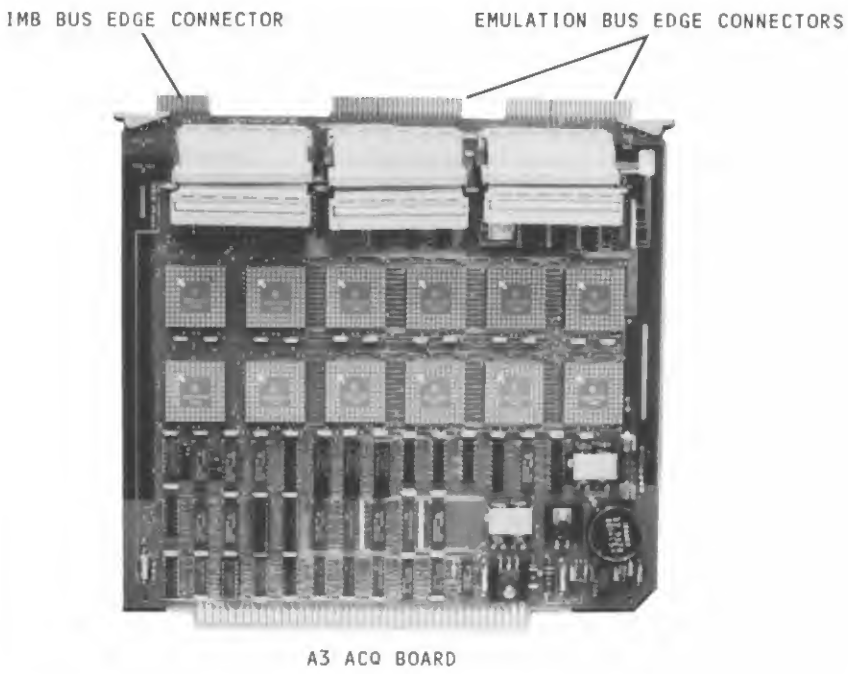


Figure 2-2. Connecting The Interconnect Cables To The Acquisition Board

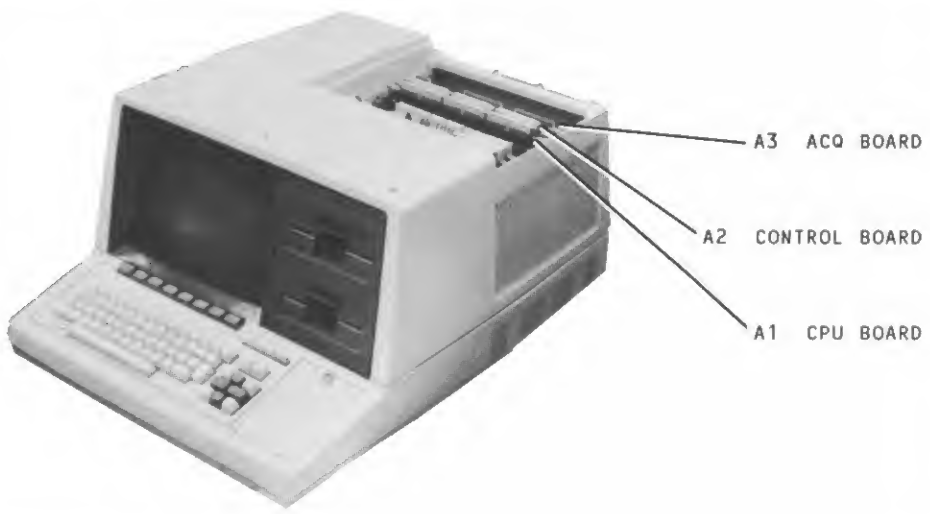
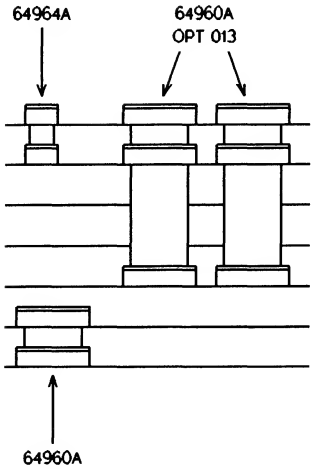
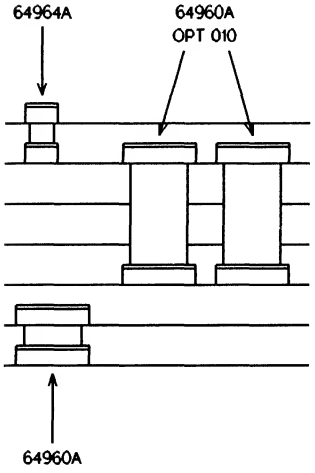


Figure 2-3. Installing The 64340A Module Into The 64100A Station

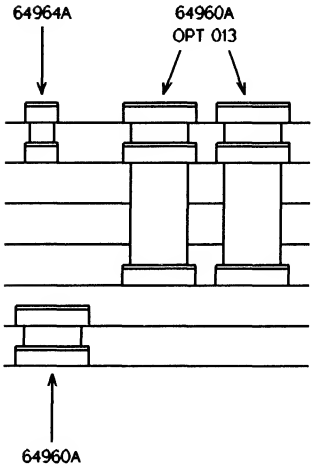
Table 2-1. HP 64340A Configurations, Current Usage, and Cable Options



		(+5 V)
SLOT 9	64302A ANL	3.70 A
8	64340A ACQ	5.00 A
7	64340A CNTL	5.00 A
6	64340A CPU	5.00 A
5	EMULATION SUBSYSTEM	≤ 6.20 A
4	64155A WIDE-ADDR MC	3.80 A
3	64161A 128K BYTE EM	1.60 A
	TOTAL CURRENT	≤ 30.30 A

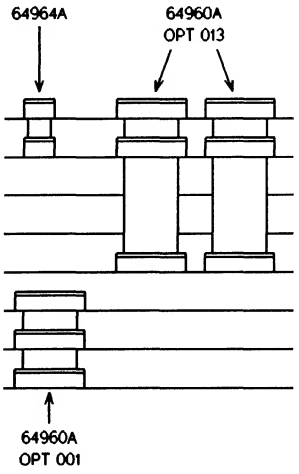


		(+5 V)
SLOT 9	64303A IMB	1.80 A
8	64340A ACQ	5.00 A
7	64340A CNTL	5.00 A
6	64340A CPU	5.00 A
5	EMULATION SUBSYSTEM	≤ 6.20 A
4	64155A WIDE-ADDR MC	3.80 A
3	64161A 128K BYTE EM	1.60 A
	TOTAL CURRENT	≤ 28.40 A

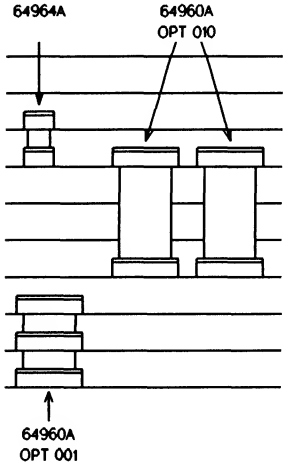


		(+5 V)
SLOT 9	64310A ANL	5.80 A
8	64340A ACQ	5.00 A
7	64340A CNTL	5.00 A
6	64340A CPU	5.00 A
5	EMULATION SUBSYSTEM	≤ 6.20 A
4	64155A WIDE-ADDR MC	3.80 A
3	64161A 128K BYTE EM	1.60 A
	TOTAL CURRENT	≤ 32.40 A

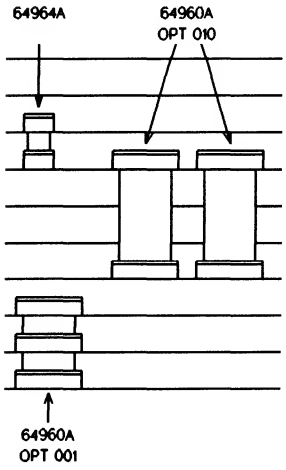
Table 2-1. HP 64340A Configurations, Current Usage, and Cable Options (Cont'd)



		(+5 V)
SLOT 9	64310A ANL	5.80 A
8	64340A ACQ	5.00 A
7	64340A CNTL	5.00 A
6	64340A CPU	5.00 A
5	EMULATION SUBSYSTEM	≤ 6.20 A
4	64155A WIDE-ADDR MC	3.80 A
3	64161A 128K BYTE EM	1.60 A
2	64161A 128K BYTE EM	1.60 A
	TOTAL CURRENT	≤ 34.00 A

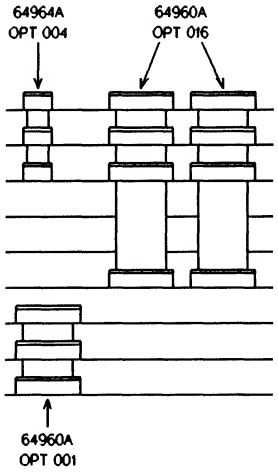


		(+5 V)
SLOT 9	OPT 011 20 CH ACQ	
8	OPT 011 40 CH ACQ	
7	64620S STATE CNTL	10.40 A
6	64340A ACQ	5.00 A
5	64340A CNTL	5.00 A
4	64340A CPU	5.00 A
3	EMULATION SUBSYSTEM	≤ 6.20 A
2	64155A WIDE-ADDR MC	3.80 A
1	64161A 128K BYTE EM	1.60 A
0	64161A 128K BYTE EM	1.60 A
	TOTAL CURRENT	≤ 38.60 A

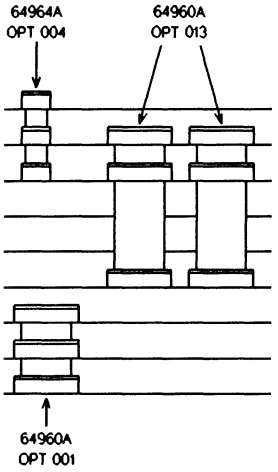


		(+5 V)
SLOT 9	OPT 012 40 CH ACQ	
8	OPT 012 40 CH ACQ	
7	64620S STATE CNTL	12.20 A
6	64340A ACQ	5.00 A
5	64340A CNTL	5.00 A
4	64340A CPU	5.00 A
3	EMULATION SUBSYSTEM	≤ 6.20 A
2	64155A WIDE-ADDR MC	3.80 A
1	64161A 128K BYTE EM	1.60 A
0	64161A 128K BYTE EM	1.60 A
	TOTAL CURRENT	≤ 40.40 A

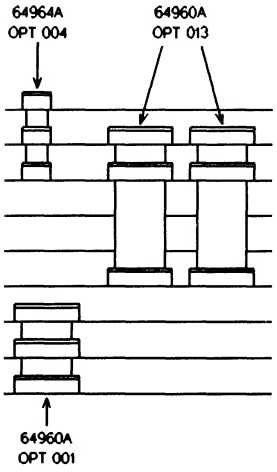
Table 2-1. HP 64340A Configurations, Current Usage, and Cable Options (Cont'd)



		(+5 V)
SLOT 9	64302A ANL	3.70 A
8	64310A ANL	5.80 A
7	64340A ACQ	5.00 A
6	64340A CNTL	5.00 A
5	64340A CPU	5.00 A
4	EMULATION SUBSYSTEM	≤ 6.20 A
3	64155A WIDE-ADDR MC	3.80 A
2	64161A 128K BYTE EM	1.60 A
1	64161A 128K BYTE EM	1.60 A
	TOTAL CURRENT	≤ 37.70 A

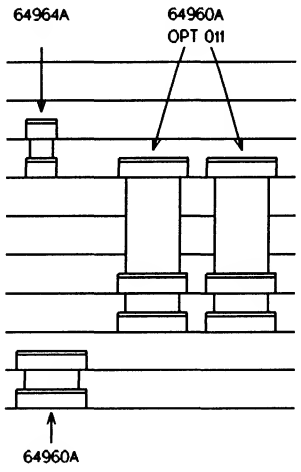


		(+5 V)
SLOT 9	64303A IMB	1.80 A
8	64310A ANL	5.80 A
7	64340A ACQ	5.00 A
6	64340A CNTL	5.00 A
5	64340A CPU	5.00 A
4	EMULATION SUBSYSTEM	≤ 6.20 A
3	64155A WIDE-ADDR MC	3.80 A
2	64161A 128K BYTE EM	1.60 A
1	64161A 128K BYTE EM	1.60 A
	TOTAL CURRENT	≤ 35.80 A

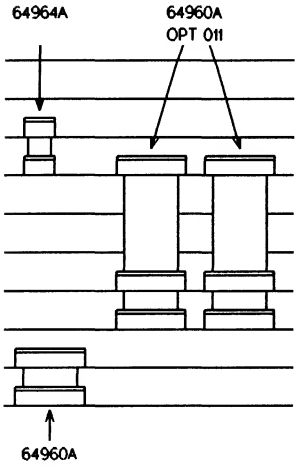


		(+5 V)
SLOT 9	64303A IMB	1.80 A
8	64302A ANL	3.70 A
7	64340A ACQ	5.00 A
6	64340A CNTL	5.00 A
5	64340A CPU	5.00 A
4	EMULATION SUBSYSTEM	≤ 6.20 A
3	64155A WIDE-ADDR MC	3.80 A
2	64161A 128K BYTE EM	1.60 A
1	64161A 128K BYTE EM	1.60 A
	TOTAL CURRENT	≤ 33.70 A

Table 2-1. HP 64340A Configurations, Current Usage, and Cable Options (Cont'd)



		(+5 V)	
SLOT 9	OPT 011 20 CH ACQ		
8	OPT 011 40 CH ACQ		
7	64620S STATE CNTL	10.40	A
6	64340A ACQ	5.00	A
5	64340A CNTL	5.00	A
4	64340A CPU	5.00	A
3	64304A EBPP	1.50	A
2	EMULATION SUBSYSTEM	≤ 6.20	A
1	64155A WIDE-ADDR MC	3.80	A
0	64161A 128K BYTE EM	1.60	A
	TOTAL CURRENT	≤ 38.50	A



		(+5 V)	
SLOT 9	OPT 012 40 CH ACQ		
8	OPT 012 40 CH ACQ		
7	64620S STATE CNTL	12.20	A
6	64340A ACQ	5.00	A
5	64340A CNTL	5.00	A
4	64340A CPU	5.00	A
3	64304A EBPP	1.50	A
2	EMULATION SUBSYSTEM	≤ 6.20	A
1	64155A WIDE-ADDR MC	3.80	A
0	64161A 128K BYTE EM	1.60	A
	TOTAL CURRENT	≤ 40.30	A

LOADING ANALYZER SOFTWARE

With your 64341 Software Analyzer, you receive three flexible discs containing all of the software analyzer software. Follow the instructions below to load this software on a clustered development system using a hard disc.

1. Press the *--BACKUP--* softkey.
2. Enter the command *floppy sys_gen* (RETURN).
3. Enter the command *copy all from local_disc <DISC #> to bus_disc <DISC#>* (RETURN).

On the left side of the display, you will see the name of the module being copied. When the module has been copied, the module name is added to the list labeled "System modules on bus disc" on the right side of the display. When all modules have been copied to the system disc, the message "*Copy complete*" is displayed on the status line. Then enter the command:

end (RETURN)

NOTE

Part of the software analyzer software is stored on the 64340A CPU board. This software is loaded to the CPU board when the software analyzer is first accessed . On subsequent accesses to the analyzer, this software may not be reloaded. To ensure that the correct software is loaded to the 64340A CPU board, always cycle power on your development station after loading new software analyzer software on your system disc.

REMOVING SOFTWARE FROM THE SYSTEM DISC

System software, such as the analyzer software, cannot be purged from the system and it cannot be removed file-by-file. System software must be removed via the *floppy sys_gen* function using the following procedure.

1. Press the *--BACKUP--* softkey.
2. Enter the command *floppy sys_gen* (RETURN).
3. Enter the command *show bus_disc <DISC #>* (RETURN).
4. Press the (NEXT PAGE) key until you locate the module you want to remove. You can remove or copy modules by their list number or by the module name.
5. Enter the command *remove <MODULE> from bus_disc <DISC #>* (RETURN) or the command *remove <NUMBER> from bus_disc <DISC #>* (RETURN).

6. When the message "*Removal complete*" is displayed on the status line, enter the command *end* (RETURN).

MAKING DUPLICATE COPIES OF FLOPPY DISC SOFTWARE

Your software analyzer was shipped on three floppy discs. You should make another copy of the floppy discs for your use and protect the original discs that you received from Hewlett-Packard. The following procedure describes how to make duplicate floppy discs so that the original discs may be stored for safekeeping.

To make a duplicate floppy disc, proceed as follows:

1. Remove a new blank floppy disc from its container and label it SOFTWARE ANALYZER. Do not write directly on the floppy disc; this can damage the floppy. Use stick-on labels, if available, or a felt-tip pen.

2. Install the original disc for the software analyzer in disc drive 0 of your HP 64000 station.

3. Install the new blank SOFTWARE ANALYZER in disc drive 1.

4. From the system monitor softkey level, press the following softkeys in the sequence shown:

   --BACKUP-- floppy utilities (RETURN)

5. The CRT display will show an explanation of the floppy utilities routines. A floppy disc must be formatted prior to use. Formatting initializes the disc, preparing it to receive information. To format the disc, press the *format* softkey and the "1" key, then press the (RETURN) key.

6. When disc 1 formatting is completed, press the *duplicate* softkey and the "0" key, then press the (RETURN) key. The contents of the software analyzer disc will be duplicated on the blank formatted disc in disc drive 1.

Perform the preceding steps for each of the software analyzer discs. This completes the procedure for making user "SOFTWARE ANALYZER" discs.

PERFORMING OPERATION VERIFICATION

Performance verification for the HP 64340A Module is a subset of the system Option Test Performance Verification. The system level PV tests all option modules that are located in the development station cardcage. You must have the software module **PV_64340** on your system disc to run performance verification on your software analyzer. This module is supplied on floppy disc with the 64340A hardware.

Procedure To Run Main Test Performance Verification:

To verify that the HP 64340A passes performance verification, perform the following:

1) Press the ---*ETC*--- softkey until the *opt_test* softkey appears.

2) Press the *opt_test* softkey, followed by the RETURN key.

3) Select one of the three HP 64340A Software Analyzer boards, and type in its card slot position, followed by the RETURN key. NOTE: It does not matter which board is selected. The same test will be executed regardless of which of the three boards is selected.

4) If IMB stimulus is present in the cardcage, the screen will ask for the IMB stimulus slot number. Type in its slot position, followed by the RETURN key.

NOTE

By selecting the HP 64340A CPU board as the IMB stimulus, the IMB tests will be bypassed.

5) Press the *cycle* softkey to test all three HP 64340A Software Analyzer boards.

6) Press the *end* softkey to stop the test and return to the *opt_test* screen.

7) Press the *end* softkey again to leave performance verification.

If any of the three tests fail, refer to the troubleshooting flowchart in the 64340A service manual.

NOTES

Chapter 3

GETTING STARTED

OVERVIEW

This chapter contains information to help you become familiar with the operation of the software analyzer. This chapter provides the following information:

- A description of the major software analyzer softkeys.
- How to setup your development system for measurements.
- How to build required database files.
- How to load and execute programs in emulation.
- How to access the software analyzer.
- How to execute a trace measurement.
- How to save a measurement configuration.
- Recommended programming style for best analysis results.

GENERAL INFORMATION

This chapter contains information to help you become familiar with the operation of the software analyzer. You will learn about the first level of analyzer softkeys and how to use them in specifying a measurement. You will learn how to build the database files required by the analyzer. You will also learn to enter measurement specifications in the software analyzer and to gather data as a result of the measurement specifications you set up. In addition, you will learn to save a configuration to a file and reload that configuration at a later time. Guidelines for writing code to achieve the best results from the software analyzer are given at the end of this chapter.

If you have any difficulties or problems when using the software analyzer, see appendix E, Resolving Measurement Problems, for possible solutions.

MAJOR SOFTKEY LEVELS

The software analyzer has a user-friendly interface designed to provide you with easily definable options to examine Pascal and C programs. The interface provides a logical structural breakdown and guided syntax softkeys that make definition of measurements easy.

The major softkey levels of the software analyzer are the ***setup***, ***display***, ***db_check***, ***modify***, ***show***, ***execute***, ***end***, ***run***, ***break***, ***reset***, ***load***, ***configure***, and ***copy*** softkeys. These softkeys are discussed briefly in the following paragraphs. This brief account of each key allows you to become familiar enough with them to perform the familiarization exercises detailed later in this chapter. A detailed explanation of all the softkeys used in, and under, the major softkey levels is given in later chapters. Syntax diagrams for the major softkey level functions are given in appendix A.

setup — The ***setup*** softkey allows you to specify; (1) the type of measurement to be made with the parameters to be traced, (2) global measurement parameters, (3) the break condition(s) to stop your program execution, (4) measurement enable and disable conditions, (5) the default path to be used in measurement specifications, i.e., the procedure and/or file information, and (6) IMB interactions.

display — The ***display*** softkey allows you to display the the current value of a specified Pascal or C variable or to modify the measurement display.

db_check — The ***db_check*** softkey allows you to check database compatibility. See chapter 4.

modify — The ***modify*** softkey enables you to modify the current value of a variable in memory. The variable can be set to a specific value which must be entered as a simple integer value less than or equal to 32 bits in width.

show — The ***show*** softkey allows you to select either the setup display, the measurement display, or the source file for display on the HP 64000 screen.

execute — The ***execute*** softkey causes execution of a measurement.

end — Pressing the ***end*** softkey one time causes the system to terminate the current measurement session and places the HP 64000 station back into the measurement system monitor. The software analyzer can be reentered from this level simply by pressing the ***sw_anl_N*** softkey. The ***end*** softkey also causes the emulation command (emul_com) file to be updated..

run — The ***run*** softkey allows you to start execution of the user program in emulation without exiting the software analyzer. When the processor is in a reset state, the ***run*** command causes the reset to release. When a ***from*** address is specified, the processor is directed to that address. If the processor is running in emulation monitor and ***real_time optional*** mode is selected, executing the single keyword ***run*** causes the processor to exit into the user program. If the software analyzer is in ***real_time required*** mode, executing the single keyword ***run*** causes the processor reset signal to be released.

break — Pressing the ***break*** softkey causes the processor to be diverted from execution of the user program to the emulation monitor. The break vector can be directed to branch the program to a user routine and continue real-time execution, without returning control to the emulator.

reset — The ***reset*** softkey allows you to suspend target system operation and reestablish initial emulator operating parameters. The reset signal is latched when active and is released by the ***run*** command.

load — The *load* softkey allows you to transfer absolute code from the HP 64000 system disc into user RAM or emulation memory. The destination of the absolute code is determined by the memory configuration map which was set up during emulation configuration and the address specified during linking.

configure — The *configure* softkey allows you to either save or load the complete analyzer configuration to or from a file.

copy — The *copy* softkey allows you to copy the measurement setup, measurement data, or the current display to either a file or the printer.

<CMDFILE> — The <CMDFILE> softkey is a prompt informing the user that a command file may be executed at this level to automatically execute software analyzer commands.

PREPARING THE SYSTEM FOR MEASUREMENTS

The information contained in this section is provided to help you become familiar with the basic operation of the software analyzer. You will be lead through the steps required to configure the HP 64000 system for performing basic software measurements. You will learn how to gain access to the analysis functions and how to setup the analyzer to make a simple trace modules measurement.

Initial Turn On

NOTE

The following procedure assumes that you have installed the HP 64340A software analyzer boards and an emulation system in your development station, and you have loaded the software analyzer software on your system disc.

1. Connect operating power to the development station.

2. Turn on the power switch. The associated indicator lamp (on HP 64110 development stations) will light.

3. You may, at this time, wish to assign a user identity code to your activity with the station. The software records your userid and assigns any files you may make to your userid. The userid must start with an upper case alphabetic character and is limited to six characters. After the first letter, the other five characters may be alphanumeric. To assign your userid press the *---ETC---* softkey twice, press the *userid* softkey, type in the userid you have selected, and press the RETURN key. If no userid is selected, the default condition is a blank userid.

   ---ETC--- ---ETC--- userid <USERID> RETURN

Building Database Files

The basis of the software analyzer measurements are the comp_db (compiler database) files. All files to be debugged with the software analyzer must have an associated comp_db file. The comp_db files allow the software analyzer to decode symbols into addresses and the addresses back into symbols. Comp_db files provide information on the symbol types (used for display purposes) and ownership of symbols by functions, procedures, or files. Comp_db files can be generated in two ways; (1) by compiling the source file with option ***comp_sym*** and linking with option ***comp_db***, or (2) by using the generate_database utility (***gen_db***).

GENERATING COMP_DB FILES AT COMPILE AND LINK TIME. The most efficient method of generating comp_db files for source files compiled on your HP 64000 Development Station is to compile the files using the ***comp_sym*** (compiler symbol) option and to link the files using the ***comp_db*** option. The following two steps build the comp_db file required by the software analyzer.

1. Compile all files that you wish to debug using the ***comp_sym*** option. This option specifies the saving of the compiler symbol file, making it available to the software analyzer. The command is:

   compile MYFILE ***options*** ... ***comp_sym*** (RETURN)

   The "..." located between ***options*** and ***comp_sym*** signifies that other options may be specified in addition to the ***comp_sym*** option. However, the ***comp_sym*** option must be the last in the list of options.

   The compiler symbol file includes the following information; the processor for which the file was compiled, the language the file was written in, the names, addresses, and data sizes for modules, and the names, types, sizes and locations of variables unique to each module. The compiler symbol file is not automatically saved after each compilation.

   An asmb_sym (assembler symbol) file is created for every file compiled unless the ***nocode*** option is included in the compile command. The contents of the asmb_sym file for a compiled file include local symbol names and relocatable or absolute addresses for those local symbols. Also, the addresses for line numbers are recorded here. In order for the analyzer to execute correctly, the asmb_sym file must be created for each file to be analyzed.

   NOTE

   DO NOT compile the file to be analyzed with the option ***nocode***. This suppresses the creation of an assembly symbol file, a file required for proper operation of the software analyzer.

2. Next, the compiled files must be linked. The command is:

   link LINK_COM_FILE ***options*** ... ***comp_db*** (RETURN)

   The link_sym (linker symbol) file is created during the linking process and contains information about all files included in the link command. Included are global symbol names and their relocated addresses, source names and their relocated addresses, and a list of memory space used by the linked files.

A database file is created at link time, when ***options comp_db*** is specified, for each file that was compiled with the ***comp_sym*** option. The ***comp_db*** must be the last specified option in the link command, as ***comp_sym*** is in the compile command.

GENERATING COMP_DB FILES USING THE GENERATE_DATABASE UTILITY. The generate_database utility allows you to generate a comp_db file for files developed in a hosted environment using HP 64000 series hosted compilers. The utility also allows you to generate comp_db files for source files developed on an HP 64000 development station, but not compiled with the comp_sym option or linked with the comp_db option. The generate_database utility provides the necessary link for performing high level software analysis in an HP 64000 development station of programs developed in the hosted development environment.

Files Required By The Generate_Database Utility. The generate_database utility requires the following files to be downloaded from the hosted development environment:

Pascal and C source files
Absolute files (.X)
Asmb_sym files (.A)
Link_sym files (.L)

A high level debug files transfer utility is available on the hosted system. This utility transfers all files required by the generate_database utility. See the Hosted Development System User's Guide for detailed information on the transfer utility.

Executing the Generate_Database Command. Executing the following command generates a comp_db file for source file MY_FILE:

generate_database MY_FILE ***using*** LINK_SYM_FILE

Where LINK_SYM_FILE is a valid link_sym file and MY_FILE is a source file referenced in the link_sym file.

This command first executes pass 1 of the compiler to generate the required comp_sym file. It then uses the asmb_sym, comp_sym, and link_sym files to generate the comp_db file. If a valid comp_sym file exists, then the following command may be executed:

generate_database comp_db MY_FILE ***using*** LINK_SYM_FILE

This command uses the existing comp_sym file, eliminating compiler pass 1 execution.

Before attempting to use the software analyzer, read chapter 4, Building Database Files. This chapter contains important information on compiling and linking files for analysis.

Loading And Executing A Program In Emulation

When the HP 64000 station is turned on, the softkey label line is displayed on the screen and contains the ***meas_sys*** softkey label. When you press the ***meas_sys*** softkey and the RETURN key, the ***sw_anl_N*** softkey will appear on the softkey label line, along with the name(s) of any emulation system in the development station. This is the measurement system level of softkeys. From here, you will need to enter the emulation system so that you can create an emulation command file

which you will need to use the software analyzer. Refer to the emulator operating manual for detailed instructions on creating emulation command files.

Selecting The Emulation Analysis Mode (64243,64245 Emulators only)

The emulator analysis mode must be set to *bus_cycle_data* in order to use the software analyzer. From within the emulation subsystem, execute the command *modify analysis_mode_to bus_cycle_data*. If the emulation analysis mode is not set to *bus_cycle_data*, the error message "**Incorrect analysis bus mode for this analyzer**" is displayed on the status line when you attempt to access the software analyzer.

Accessing The Software Analyzer

After you have generated an emulation command file in an emulation session, you are ready to access the software analyzer. Leave the emulation system running and press the *end* softkey and the RETURN key. This will bring you out to the measurement system level of softkeys. You can now access the software analyzer by pressing the *sw_anl_N* softkey, entering the name of the emulation command file, and pressing RETURN.

sw_anl_N <EMUL_COM_FILE> RETURN

If you omit the emulation command file in the command line, the software analyzer prompts you for the file;

Emulation command file?

After entering the name of the emulation command file you generated in the emulation session and pressing RETURN, you access the software analyzer, ready to start your analysis session.

NOTE

On first accessing the software analyzer, you must specify an emulation command file. During the analysis session, you can save the measurement setup in a configuration file. On subsequent uses of the software analyzer, you can specify either a configuration file or an emulation command file in the *sw_anl_N* command.

Figure 3-1 shows the utility softkeys used to gain access to the software analyzer and how to end out of the analyzer and return to the system monitor level of softkeys. Pressing the *end* softkey followed by RETURN once will return you to the measurement system level of software. The software analyzer will retain its current measurement setup. To go to the system monitor level of software press the *end* softkey again. It is now possible to perform operations at this level (edit, copy, etc.). To reuse the analyzer, and still retain the current measurement setup, press the *meas_sys* softkey and *continue* softkeys, then the RETURN key. This brings you to the measurement system level of software. Now press the *sw_anl_N* softkey and then the RETURN key. You are now back in the software analyzer with the current measurement setup retained.

NOTE

Pressing the *sw_anl_N* softkey when the measurement system is entered with the continue option restores the last measurement setup used in the software analyzer if that session was terminated using the ***end*** command and the emulator hardware is in the same state. The emulator hardware will be in the same state it was left in provided that: (1) the HP 64000 station has not been turned off, (2) the emulator has not been modified during an emulation session, or (3) ***opt_test*** has not been executed.

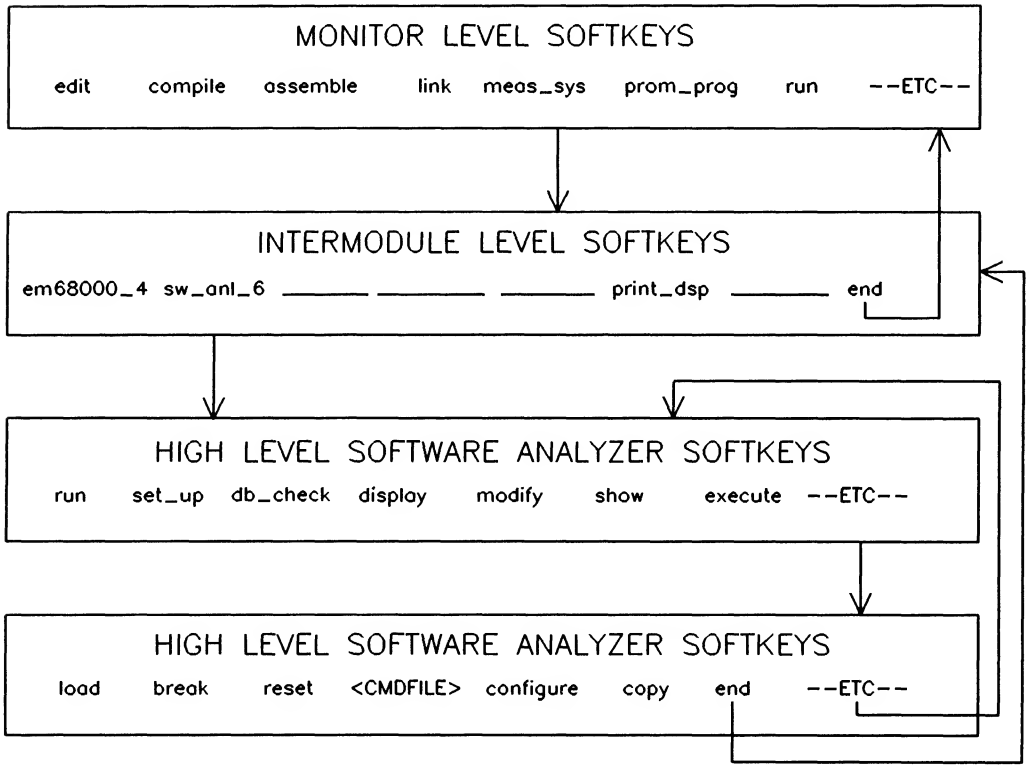


Figure 3-1. Utility Keys Used To Access the Analyzer

PERFORMING A BASIC TRACE MODULES MEASUREMENT

The following measurement example is intended to familiarize you with the software analyzer as well as show a meaningful measurement on a program written in Pascal. Figure 3-2 shows an outline of the program used in the following example. The program listing shows the procedure declarations and entry and exit points of the procedures traced in the example measurement.

Loading And Running A Program

The first step in preparing to make a measurement is to load the absolute file we wish to analyze into emulation memory. This source file must have been compiled using the *comp_sym* option and the absolute file linked using the *comp_db* option as explained previously. These two options create the symbolic data base required by the software analyzer to interpret and display measurement data. The command is executed by pressing the *load* softkey, typing in the absolute file name, and pressing the RETURN key.

load **MYFILE** RETURN

The next step is running the program in emulation. Entering the command

run at_execution from transfer_address RETURN

causes the user program MYFILE to begin running from its starting address when a measurement is executed. The *at_execution* parameter is included here because we wish to ensure that we trace all modules executed from the beginning of the program. Leaving out the *at_execution* parameter causes the user program to begin running immediately.

Defining A Default Path (Optional)

The next step in making a measurement with the software analyzer is defining the default path. The default path may be a module within a file or a file itself. The default path is used by the software analyzer when a command requires a path definition, but none is included in the command statement itself. For this measurement example, the default path is defined as the file MYFILE using the command:

setup default_path **MYFILE** RETURN

Therefore, for any commands being executed that do not include a file specification, the software analyzer will look for the defined parameters in the default path file MYFILE.

```
 1 00000000  1  "68000"
 6 00000000  1  $WARN+$
 7 00000000  1  $EXTENSIONS ON$
 8 00000000  1  PROGRAM TESTP;
 ...
13 00000000  1  TYPE
14 00000000  1  INT            = SIGNED_16;
15 00000000  1  PTR            =^INT;
16 00000000  1  SCALAR_TYPE    =(BLACK,BROWN,RED,ORANGE,YELLOW,GREEN,BLUE,VIOLET,GREY,WHITE);
17 00000000  1  DAY_OF_WEEK    =(SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY);
18 00000000  1  SUBRANGE_TYPE  =RED..YELLOW;
19 00000000  1  SET_TYPE       =SET OF SCALAR_TYPE;
20 00000000  1  ARRAY_TYPE0    =ARRAY[DAY_OF_WEEK] OF SCALAR_TYPE;
21 00000000  1  ARRAY_TYPE1    =ARRAY[SUBRANGE_TYPE] OF SCALAR_TYPE;
22 00000000  1  ARRAY_TYPE2    =ARRAY[-3..-1] OF BYTE;
23 00000000  1  ARRAY_TYPE3    =ARRAY[0..1] OF ARRAY_TYPE2;
24 00000000  1  REC_TYPE_PTR   =^REC_TYPE;
25 00000000  1  REC_TYPE       =RECORD
26 00000000  2    I            :SIGNED_32;
27 00000000  2    REAL_NUMBER  :REAL;
28 00000000  2    CHAR1        :CHAR;
29 00000000  2    FLAG         :BOOLEAN;
30 00000000  2    SETT         :SET_TYPE;
31 00000000  2    NEXT_REC     :REC_TYPE_PTR;
32 00000000  2
33 00000000  2    CASE N       :BYTE OF
34 00000000  2         1:      (VARIANT1  :ARRAY_TYPE1);
35 00000000  2         2:      (VARIANT2  :ARRAY_TYPE2);
36 00000000  2         3:      (VARIANT3  :ARRAY_TYPE3);
37 00000000  1    END;
 ...
42 00000000  1  VAR
43 00000000  1      COLOR     :ARRAY_TYPE0;
44 00000008  1      A,B       :ARRAY_TYPE1;
45 00000010  1      C         :ARRAY_TYPE2;
46 00000014  1      DAY       :DAY_OF_WEEK;
47 00000015  1      E,F       :ARRAY_TYPE3;
48 00000026  1      NEXT_COLOR:SCALAR_TYPE;
49 00000027  1      Q,R       :REC_TYPE;
50 0000005C  1      J,K       :BYTE;
51 0000005E  1      COUNT     :INTEGER;
52 00000062  1      X         :INT;
53 00000064  1      Y         :PTR;
54 00000068  1
55 00000068  1  $PAGE$
 ...
```

Figure 3-2. Listing of Example Pascal Program

```
 67 00000000  1  PROCEDURE INITHEAP(START,LENGTH:INTEGER);EXTERNAL;
  ...

 72 00000000  1  PROCEDURE PROC1 (VAR FPARM1:INTEGER; FPARM2:INTEGER);
  ...

 83 00000000  2      PROCEDURE RECURSIVE_PROC(VAR RP1:INTEGER; RP2:INTEGER;
 84 00000000  3                  VAR RP3:INTEGER; RP4:INTEGER; VAR RP5:INTEGER;
 85 00000000  3                  RP6:INTEGER; VAR PTR:REC_TYPE_PTR);
  ...

 98 00000004  3      BEGIN  (* RECURSIVE_PROC ENTRY *)
  ...

120 00000074  3          RECURSIVE_PROC (RP1,RP2,RP3,RP4,RP5,RP6,PTR);
  ...

132 0000015C  3      END;   (* RECURSIVE_PROC EXIT *)
  ...

135 00000164  2  BEGIN  (* PROC1 ENTRY *)
  ...

145 000001A6  2      RECURSIVE_PROC (FPARM1, FPARM1, FPARM2, FPARM2, D, D,
146 00000000  2                      REC_PTR^.NEXT_REC);
  ...

150 00000204  2  END;   (* PROC1 EXIT *)
  ...

153 00000000  1  PROCEDURE PROC2 (A:INTEGER);
  ...

160 00000000  2      PROCEDURE NESTED_PROC (PARM:INTEGER);
161 0000020C  3
162 0000020C  3      BEGIN (* NESTED_PROC *)
  ...

164 00000210  3      END;  (* NESTED_PROC *)
  ...

166 00000218  2  BEGIN (* PROC2 ENTRY *)
167 00000218  2      NESTED_PROC (A);
168 00000224  2  END;  (* PROC2 EXIT *)
  ...

190 0000023C  1  BEGIN  (* MAIN PROGRAM ENTRY *)
  ...

230 00000300  1      PROC1 (COUNT,COUNT+2);
231 00000312  1      PROC2 (COUNT+2);
  ...

242 00000390  1  END. (* MAIN PROGRAM EXIT *)
```

Figure 3-2. Listing of Example Pascal Program (Cont'd)

Setting Up The Trace Specification

The last step remaining before executing a trace measurement is setting up the trace specification. Entering the command

setup trace modules all RETURN

will cause the software analyzer to be configured to trace all modules in the default path file MYFILE. Had we wished to trace modules in a file other than the default path, we could have by adding the *file* parameter followed by the file name to the *setup* command. We have now set up the measurement and the complete setup display is shown in figure 3-3.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

TRACE MODULES

     module          file
     all             MYFILE:TESTP

RUN_AT_EXECUTION from
     transfer_address

DEFAULT_PATH
     file MYFILE:TESTP

REAL_TIME
     optional

COUNTER
     counts_time

STATUS: Database search successful                                    16:19

 setup  trace  modules  all


  run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 3-3. Software Analyzer Setup Display

Interpreting The Trace Listing

Pressing the *execute* softkey followed by RETURN causes the software analyzer to initiate the trace modules measurement and start execution of the user program. After the trace memory is filled, the measurement stops, and the acquired data is processed and displayed on the screen. The trace modules listing is shown in figure 3-4.

The symbol field contains the names of the modules traced. In this example, all modules are Pascal procedures. The software analyzer looks up the module name in the compiler symbol file corresponding to the traced address value in the data record and displays that symbol in the display symbol field. The status field shows whether the traced address is the entry point to the module or the exit point from the module. The time-rel field shows the time between execution of the first state of a line and execution of the first state of the preceding line. To display the source field, the software analyzer looks up the source file line number contained in the assembler symbol file and extracts that line from the source file for display. The number "36" displayed in the status line indicates that the current trace line (displayed in inverse video in the center of the display) corresponds to acquisition state 36.

Looking at the program listing in figure 3-2, we see that the main program begins at line 190. The first module traced is PROC1, being called at line 230. The procedure INITHEAP defined at line 67 is a 68000 library function and is not included in the compiler symbol file for file MYFILE. Execution of PROC1 begins at line 135 whereas the line number displayed in the listing is 230, the line from which PROC1 was called. The called address of a module is considered the entry point.

The second line in the trace listing shows RECURSIVE_PROC being called at line 145. Note that RECURSIVE_PROC is indented one column, indicating that it is called from within PROC1. Nesting of modules is indicated by indentation. We see successive calls to RECURSIVE_PROC, each indented one column from the other, followed by successive exits from the module. This indicates that RECURSIVE_PROC is a recursive routine. This is verified by the program listing in figure 3-2. After execution of RECURSIVE_PROC, the program exits PROC1 and the main program then calls PROC2. PROC2, in turn, calls NESTED_PROC.

In this manner, the software analyzer provides an overview of program activity that enables you to quickly determine whether the program is executing modules in the sequence intended or, if not, in which module the program is in error. In the later case, the user can now use other software analyzer measurements to isolate the error more precisely.

SAVING THE CONFIGURATION

If you wish to retrieve the measurement setup for use at a later time, you need to save it in a configuration file. In this way you can begin to build a library of configurations and save a great deal of time in future measurement sessions. Pressing the *configure* and *save_in* softkeys in the sequence shown, typing in CONFIG1 and pressing the RETURN key will save the present configuration in a file named CONFIG1.

configuration save_in **CONFIG1** RETURN

This allows you to change your configuration (or end the session) with the assurance that you can retrieve your current configuration at a later time, if desired.

This completes the introduction to the software analyzer. You have seen how to load and execute a program with the emulation system and how to perform a simple measurement. For more specific and detailed measurements, refer to the information contained in the following chapters.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Symbol                    Stat   Time-rel Source

PROC1                     entry  993.4 uS  230 PROC1 (COUNT,COUNT+2);
 RECURSIVE_PROC           entry  313.3 uS  145 RECURSIVE_PROC (FPARM1, FPARM1,
  RECURSIVE_PROC          entry  623.0 uS  120 RECURSIVE_PROC (RP1,RP2,RP3,RP4,
   RECURSIVE_PROC         entry  615.0 uS  120 RECURSIVE_PROC (RP1,RP2,RP3,RP4,
    RECURSIVE_PROC        entry  617.3 uS  120 RECURSIVE_PROC (RP1,RP2,RP3,RP4,
    RECURSIVE_PROC        exit   6.557 mS
   RECURSIVE_PROC         exit   6.020 mS
  RECURSIVE_PROC          exit   5.982 mS
 RECURSIVE_PROC           exit   5.932 mS
PROC1                     exit   589.2 uS
PROC2                     entry   12.3 uS  231 PROC2 (COUNT+2);
 NESTED_PROC              entry   11.2 uS  167 NESTED_PROC (A);
 NESTED_PROC              exit     5.4 uS
PROC2                     exit     5.9 uS

STATUS: Awaiting Command                                        36    16:45

   run     setup   db check   display   modify    show    execute   ---ETC---
```

Figure 3-4. Trace Modules Measurement Display

RECOMMENDED PROGRAMMING STYLE

The following programming style suggestions are recommended to achieve the best results from your analysis session.

1. Put only one statement on each line, especially when variables are used in more than one statement. The analyzer cannot determine which access has been made and only the first one may be displayed.

2. Break up compound statements, such as "IF <exp> THEN <stmt> ELSE <stmt>" to at least one line for each of the three parts.

3. Put comments on all "END" text to indicate to which structure it belongs. e.g. "END; /*FOR count LOOP*/"

4. Use BEGIN/END pairs on separate lines to mark all control structures and statements. This is redundant information in terms of compiler semantics and produces no additional code, but it clarifies the source display in the measurement analysis.

5. Put subroutine calls with all parameters on one line when possible.

Chapter 4

BUILDING DATABASE FILES

OVERVIEW

This chapter provides the following information:

- A description of database files.
- How to build database files at compile and link time.
- How to build database files using the generate_database command.
- Detailed command syntax for the generate_database command.
- How to verify database files.
- Effects of using compiler directives.
- How to trace variables within an assembly language module.

GENERAL INFORMATION

The software analyzer has a high level of interaction with the HP 64000 compilers. This chapter describes the symbolic interface between the analyzer and compiler, and how the analyzer database is built when a program is compiled and linked. It also describes how to verify that the database file is correct. A list of compiler directives and the implications of their use with the software analyzer is also discussed.

SYMBOLIC INTERFACE

The software analyzer provides the capability for the user who has developed programs using the HP 64000 Logic Development System compilers, assemblers, and linker to specify measurements in terms of the symbols used in the programs. The compilers, assemblers and linker produce symbol tables that provide the analyzer with the information necessary to determine the physical addresses associated with the user's symbols.

The software analyzer accommodates both statically stored symbols (global variables and the names of software modules such as programs, functions, and procedures) and dynamically stored symbols (local variables, VAR parameters, and value parameters). The analyzer also allows you to reference source statement line numbers. The different symbol storage classifications and data types are explained in detail in chapter 15.

COMP_DB FILES

The basis of the software analyzer measurements are the comp_db (compiler database) files. The comp_db files allow the software analyzer to decode symbols into addresses and the addresses back into symbols. The comp_db files provide information on the symbol types (used for display purposes) and ownership of symbols by functions, procedures, or files.

Comp_db files can be generated in two ways; (1) by compiling the source file with option ***comp_sym*** and linking with option ***comp_db***, or (2) by using the ***generate_database*** utility. Due to the time required to build the comp_db files, it is suggested that you keep only a small working set of these files.

Since the linker creates a comp_db file for each comp_sym (compiler symbol) file it finds in the list of files being linked, old comp_sym files that are not being used should be purged. For each comp_sym file purged, the corresponding comp_db file should also be purged so that the software analyzer will not use a file that is not up-to-date. Conversely, when you make changes to files which are being tested and fail to either compile with the ***comp_sym*** option or link with the ***comp_db*** option, then unpredictable results can occur in the software analyzer measurements The software analyzer has a ***database_check*** command that allows you to verify that all comp_db files are up-to-date. The ***database_check*** command is described later in this chapter.

BUILDING THE DATABASE FILE

The procedure for building the symbol database required by the software analyzer is described in the following paragraphs. The procedure is illustrated graphically in figure 4-1.

Compiling Files

All files that you wish to debug must be compiled with the ***comp_sym*** option which specifies the saving of the compiler symbol file. The command is:

compile <filename> options ... comp_sym

The "..." located between ***options*** and ***comp_sym*** signifies that other options may be specified in addition to the ***comp_sym*** option. However, the ***comp_sym*** option must be the last in the list of options.

COMPILER SYMBOL FILE. A compiler symbol file is generated for each file compiled on the HP 64000 system. The compiler symbol file includes the following information; the processor for which the file was compiled, the language the file was written in, the names, addresses, and data sizes for modules, and the names, types, sizes and locations of variables unique to each module. The compiler symbol file is not automatically saved after each compilation. The ***comp_sym*** option must be specified at compilation time which causes the compiler symbol file to be saved, making it accessible to the software analyzer.

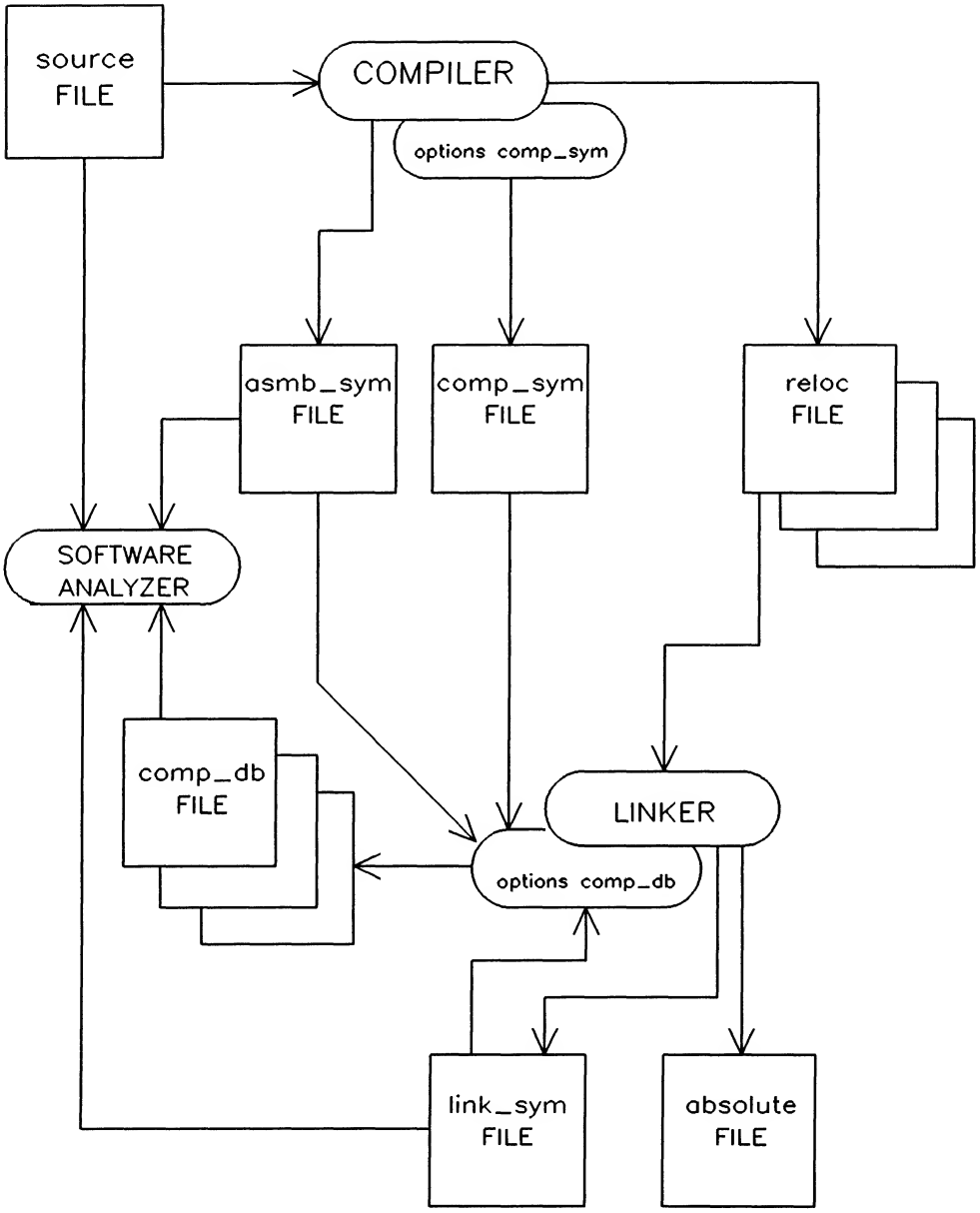


Figure 4-1. Software Analyzer Symbolic Interface

ASSEMBLER SYMBOL FILE. An asmb_sym (assembler symbol) file is created for every file compiled unless the ***nocode*** option is included in the compile command. The contents of the asmb_sym file for a compiled file include local symbol names and relocatable or absolute addresses for those local symbols. Also, the addresses for line numbers are recorded here. In order for the analyzer to execute correctly, the asmb_sym file must be created for each file to be analyzed.

NOTE

DO NOT compile the file to be analyzed with the option ***nocode***. This will suppress the creation of an assembly symbol file, a file required for proper operation of the software analyzer.

Linking Files

Next, the compiled files must be linked. The command to use is:

link ... options ... comp_db

The link_sym file is created during the linking process and contains information about all files included in the ***link*** command. Included are global symbol names and their relocated addresses, source names and their relocated addresses, and a list of memory space used by the linked files.

A compiler data base (comp_db) file is created at link time, when ***options comp_db*** is specified, for each file that was compiled with the ***comp_sym*** option. The ***comp_db*** must be the last specified option in the link command, as ***comp_sym*** was in the compile command.

NOTE

When Pascal and C files are linked within the same absolute file, the linker will execute faster if the Pascal and C files are linked in separate blocks and not intermixed with each other.

Using The Generate_Database (gen_db) Command.

The generate_database Command allows you to generate comp_sym and comp_db type files without the overhead of recompilation and relinking. These files are optionally generated by the HP 64000 hosted compilers and linkers. The generate_database utility provides the capability to generate the comp_sym and comp_db type files for source files developed in HP supported hosted environments other than the HP 64000 environment. The generate_database utility provides the necessary link for performing high level software analysis in an HP 64000 development station of programs developed in the hosted development environment.

The utility can also be used to generate comp_db files for source files developed on an HP 64000 development station, but not compiled with the comp_sym option or linked with the comp_db option. This command first executes pass 1 of the compiler to generate the required comp_sym file. It then uses the asmb_sym, comp_sym, and link_sym files to generate the comp_db file. If a valid comp_sym file exists, you can specify that only the comp_db file be generated. The command then uses the existing comp_sym file, eliminating compiler pass 1 execution.

Specifying the keyword ***comp_sym*** allows you to perform syntax checking on a source program without the overhead of compiling the program.

NOTE

You do not need a compiler on your HP 64000 system to generate comp_sym and comp_db files using the *generate_database* command.

REQUIRED FILES. The generate_database utility requires the following files to be downloaded from the hosted development environment:

Pascal and C source files
Absolute files (.X)
Asmb_sym files (.A)
Link_sym files (.L)

A high level debug files transfer utility is available on the hosted system. This utility transfers all files required by the generate_database utility. See the Files Transfer Utilities section of the Hosted Development System User's Guide for detailed information on the transfer utility.

GENERATE_DATABASE COMMAND SYNTAX. The command syntax for the generate_database command is shown in figure 4-2.

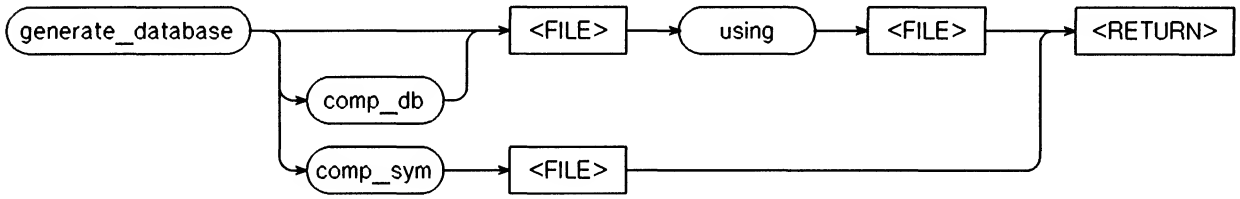


Figure 4-2. Generate_Database Command Syntax Diagram

GENERATE_DATABASE COMMAND PARAMETERS. The following parameters can be specified in the generate_database command.

comp_db — ***comp_db*** specifies that only a comp_db file be generated.

comp_sym — ***comp_sym*** specifies that only a comp_sym file be generated.

<FILE> — **<FILE>** must be the name of a C or Pascal source file for which the comp_sym and/or comp_db files are to be generated. In order for the database generator to distinguish between the two, C source files must have "C" for their first line and Pascal files must have "PASCAL" for their first line.

using <FILE> — ***using*** **<FILE>** specifies the link_sym file (generated by a previous link) to be used in generating the comp_db file.

GENERATE_DATABASE COMMAND EXAMPLES. The following command examples illustrate how to use the generate_database command.

generate_database C_SOURCE *using* C_LINKSYM

The files C_SOURCE:comp_sym and C_SOURCE:comp_db are generated. C_SOURCE is a C source file and C_LINKSYM is a link_sym file generated by a previous link of C_SOURCE with other relocatables.

generate_database comp_sym PASCAL_SOURCE

The file PASCAL_SOURCE:comp_sym is generated.

generate_database comp_db PASCAL_SOURCE *using* PASCAL_LINKSYM

The file PASCAL_SOURCE:comp_db is generated.

VERIFYING DATABASE FILES

For proper operation of the software analyzer, the database information provided by the compiler and linker must be current for the files being analyzed. The *database_check* command is provided to systematically verify whether the database files associated with the current absolute file are up-to-date. The following paragraphs describe how database files are verified. The *database_check* command syntax is shown in figure 4-3.

The normal sequence of creating files is described in the previous section of this chapter, Building the Database Files. This process generates the following files in the order listed: 1) the comp_sym file, 2) the reloc file, and 3) the asmb_sym file. The reloc files are then linked with option *comp_db* specified. This generates, in order, the link_sym, absolute, and comp_db files.

During the normal operation of a link, the link_sym file will always be dated before or with the same date and time as the absolute file. The database_check function compares the modify date of the current absolute file with the modify date of the link_sym file associated with that absolute file. If the absolute file's modify date and time is earlier than that of the link_sym file, a database error exists. This error will be displayed and the database check will be terminated.

Since a relocatable file may be linked to any number of absolute files, the comp_db file for each file compiled with options comp_sym will contain the file name of the last absolute file it was linked to. With this information, the database_check performs the following operations:

1. Obtains the names of all linked files from the link_sym file.

2. Determines if the named files have comp_db files (i.e., if they were compiled with option *comp_sym* and linked with option *comp_db*).

3. Verifies that comp_db files were generated for the current absolute file by comparing the name of the currently loaded absolute file with the absolute file name contained in the comp_db files.

4. Verifies that all files were generated in the proper sequence by comparing the modify date and time for each file type.

5. Reports any discrepancies in the database.

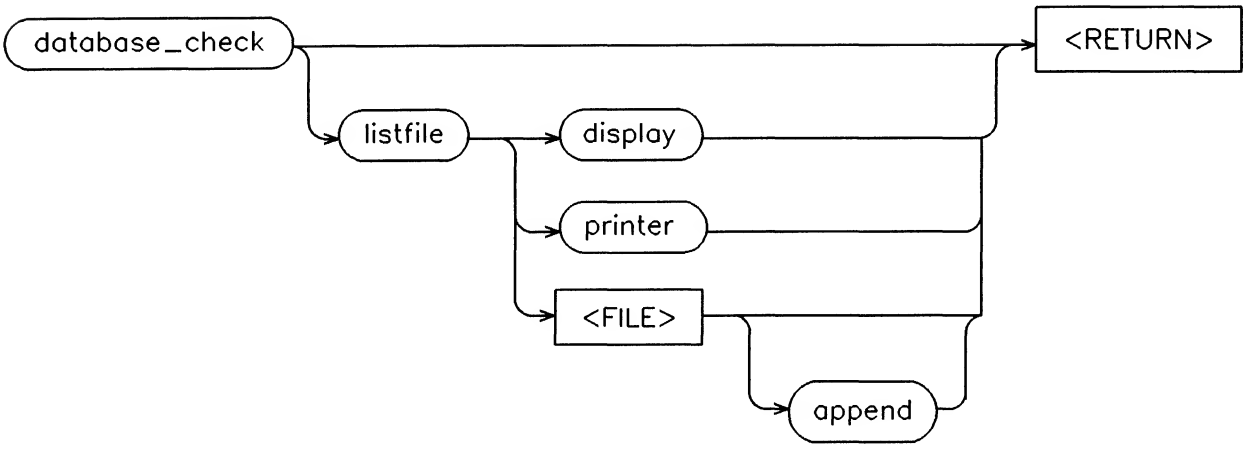


Figure 4-3. Database_check Command Syntax Diagram

The following example commands illustrate use of the ***database_check*** command:

database_check listfile display
database_check listfile printer
database_check listfile DATABASE_CHECK ***append***

NOTE

To ensure that each file has the correct modify date, always keep your system clock set to the current date and time. This is done using the ***date&time*** utility command at the system monitor level of softkeys.

USING COMPILER DIRECTIVES

There are certain compiler directives that must be in the ON state for the Software Analyzer to operate correctly and others that may cause unexpected results.

AMNESIA

When the ***AMNESIA*** option is ***OFF***, there may be accesses to variables that could be missed because they are stored in registers. The default value is ***OFF***.

ASMB_SYM

ASMB_SYM must be ***ON***. The default value is ***ON***.

FIXED_PARAMETERS (C only)

When *FIXED_PARAMETERS* is *OFF* the software analyzer may display a parameter as being accessed when it was not. This occurs when the calling routine does not pass all the parameters to the called routine. The default value is *OFF*.

LINE_NUMBERS

LINE_NUMBERS must be *ON*. The default value is *ON*.

OPTIMIZE

If the *OPTIMIZE* option is *ON*, some accesses to variables may be missed because they are stored in registers. The default value is *OFF*.

FILES WRITTEN IN ASSEMBLY LANGUAGE

If a module is written in assembly language and you wish to trace a variable within the assembly language module, the variable must be declared as external in some other Pascal or C file. When the assembly language variable is not declared external, its address will not appear in the data base of any file and the analyzer will not be able to find the variable when a measurement is specified.

Chapter 5

DEFINING MEASUREMENT PARAMETERS

OVERVIEW

This chapter describes how to define the following measurement parameters.

- default_path
- counter
- real_time
- absolute_file
- trigger_enable

INTRODUCTION

The ***setup*** command allows you to define several parameters that affect how the software analyzer performs measurements. These parameters are ***default_path***, ***counter***, ***real_time***, ***absolute_file***, and ***trigger_enable***.

default_path

default_path specifies the software path to be used by the software analyzer when a command requires a path definition, but none is included in the command statement itself. The default path may be a procedure or function (module) within a source file or the source file itself. Using *default_path* to specify a particular segment of your software can simplify the setup commands used during a measurement session. When path parameters (proc and file, or file) are omitted from a command, the default path is used to locate variables and procedures in the program under test.

Syntax

The command syntax for setting up a default path is shown in figure 5-1.

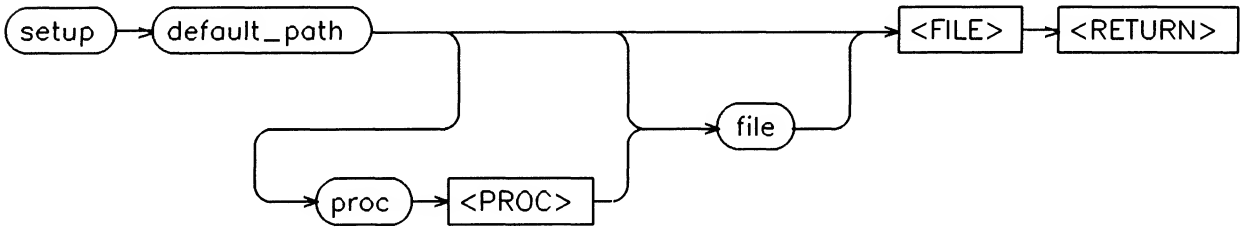


Figure 5-1. Setup Default_Path Command Syntax

Default Value

none

Parameters

file	*file* indicates that the name of a source file follows. **NOTE**: A colon (:) may be used in place of pressing the *file* softkey.
<FILE>	<FILE> specified the source file to be used as the default path.
proc	*proc* indicates that a procedure or function name follows that defines the procedure or function to which a variable belongs. **NOTE**: an "@" may be use in place of *proc*.
<PROC>	<PROC> is an optional parameter that specifies a procedure or function within a source file as the default_path. If <PROC> is defined in the *setup default_path* command, it may be omitted in the measurement command line. If <PROC> is not specified in either the default path or the measurement command line, the analyzer assumes that any specified variables are global variables defined at the main program level.

default_path (Cont'd)

Examples

setup default_path proc SORT_ELEMENTS *file* SORT
setup default_path file MATRIX

counter

counter allows you to define how the software analyzer time-state counters are used. ***to_count_state*** specifies that the counters count state transactions (bus cycles). ***to_count_time*** specifies that the counters count time.

Syntax

The command syntax for setting up the counters is shown in figure 5-2.

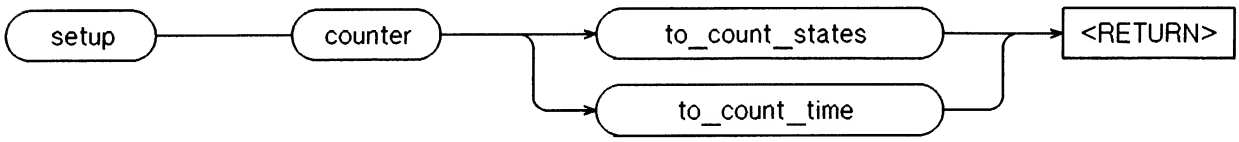


Figure 5-2. Setup Counters Command Syntax

Default Value

count_time

Parameters

to_count_states	***to_count_states*** specifies that the hardware counter is to count bus cycles.
to_count_time	***to_count_time*** specifies that the hardware counter is to count time.

Examples

setup counter to_count_states
setup counter to_count_time

real_time

The *real_time* parameter allows you to specify whether or not the software analyzer is allowed to break to the emulation monitor during a measurement. The parameter also determines whether or not the emulator can be halted during a run to check emulator status. *real_time optional* allows the analyzer to break to the emulator monitor or to halt the emulator during a measurement. *real_time required* specifies that the analyzer cannot break to the emulator monitor or halt the emulator during a measurement. Specifying *real_time required* limits the type and quantity of measurements that the analyzer can perform. See the detailed measurement descriptions for information on how *real_time* affects specific measurements.

Syntax

The command syntax for defining the real_time parameter is shown in figure 5-3.

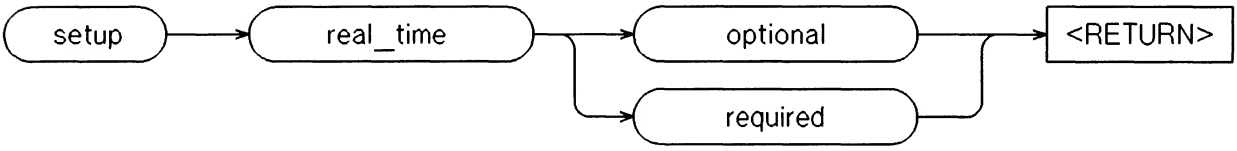


Figure 5-3. Setup Real_time Command Syntax

Default Value

Value is taken from the emulation command file.

Parameters

optional	*optional* allows the analyzer to break to the emulator monitor.
required	*required* prohibits the analyzer from breaking to the emulator monitor.

Examples

```
setup real_time optional
setup real_time required
```

absolute_file

The *absolute_file* parameter allows you to specify an absolute file to be traced in the event that no absolute file is loaded via any HP 64000 analysis subsystem. For example, this command could be used to specify a file that is stored in ROM installed in the target system. This command is also used to specify one absolute file to be traced when multiple files are loaded into emulation memory.

Syntax

The command syntax for specifying an absolute file is shown in figure 5-4.

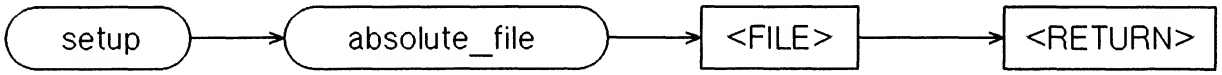


Figure 5-4. Setup Absolute_file Command Syntax

Default Value

none if no *setup absolute_file* or *load* command has been executed, otherwise the last file loaded.

Parameters

<FILE>	<FILE> is the name of the absolute file to be traced by the software analyzer.

Example

setup absolute_file MATRIX

trigger_enable

trigger_enable is used with the *setup* command to define the IMB interaction between the software analyzer and other measurement subsystems installed in the HP 64000 development station. The software analyzer must be in *real_time required* mode in order to interact with the Intermodule Bus (IMB).

Syntax

The command syntax for specifying the *trigger_enable* condition is shown in figure 5-5.

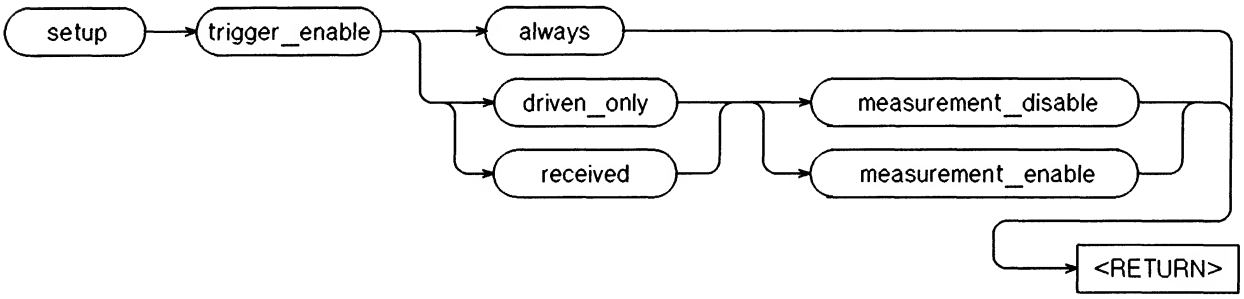


Figure 5-5. Setup Trigger_Enable Command Syntax

Default Value

always

Parameters

always — *always* specifies that *trigger_enable* is always true. This, in effect, removes the analyzer from the IMB (Intermodule Bus).

driven_only — *driven_only* specifies that the IMB trigger_enable line is to be driven on measurement_enable or measurement_disable.

measurement_disable — *measurement_disable* specifies that the IMB trigger enable line is to be driven when the specified measurement disable condition is true or received to initiate looking for the measurement disable condition. If no measurement disable condition is specified, the trigger enable is ignored.

measurement_enable — *measurement_enable* specifies that the IMB trigger enable line is to be driven when the specified measurement enable condition is true or received to initiate looking for the measurement enable condition. If no measurement enable condition is specified, the trigger enable is ignored.

trigger_enable (Cont'd)

received — *received* specifies that the analyzer measurement will start looking for the measurement disable or enable condition when the IMB trigger enable line is set true by another HP 64000 measurement subsystem.

Examples

```
setup trigger_enable always
setup trigger_enable driven_only measurement_disable
setup trigger_enable received measurement_enable
```

Chapter 6

QUALIFYING MEASUREMENTS

OVERVIEW

This chapter describes the following measurement functions used to qualify measurement data acquisition.

- measurement_enable
- measurement_disable
- windowing

GENERAL INFORMATION

The ***measurement_enable*** and ***measurement_disable*** commands allow you to enable or disable measurement execution on specified terms, 'OR'ed combinations of terms, or sequences of terms. A term may be a source program line number, entry to a program module, exit from a program module, or any state (used in conjunction with ***trigger_enable***). The ***measurement_disable*** command allows you to define a measurement window, enabling you to make repetitive measurements over a specified program range.

Measurement Enable

The *measurement_enable* command allows you to qualify data acquisition as shown in figure 6-1. Measurement enable is AND'ed with the measurement specification condition. This allows you to define more precisely the location in program execution that is to be traced. During measurement execution, data is acquired only when both the measurement specification is satisfied and the measurement is enabled. While both conditions are true, the software analyzer acquires data until its trace memory is filled, until a measurement disable term is encountered, or until the measurement is halted by the user. Both a full trace memory and occurrence of a measurement disable term results in measurement completion as defined for the *setup break* and *wait* commands. The measurement enable function can be used in both *real_time optional* mode and *real_time required* mode. In *real_time required* mode, the measurement enable term is displayed if it is part of the program segment being traced. In *real_time optional* mode, the measurement enable term may not be displayed on the measurement display.

Measurement enable can be viewed as a sequential term used with the measurement specification, i.e., find the measurement enable condition and then start the measurement. Once set true, measurement_enable remains true unless a specified measurement disable condition is found.

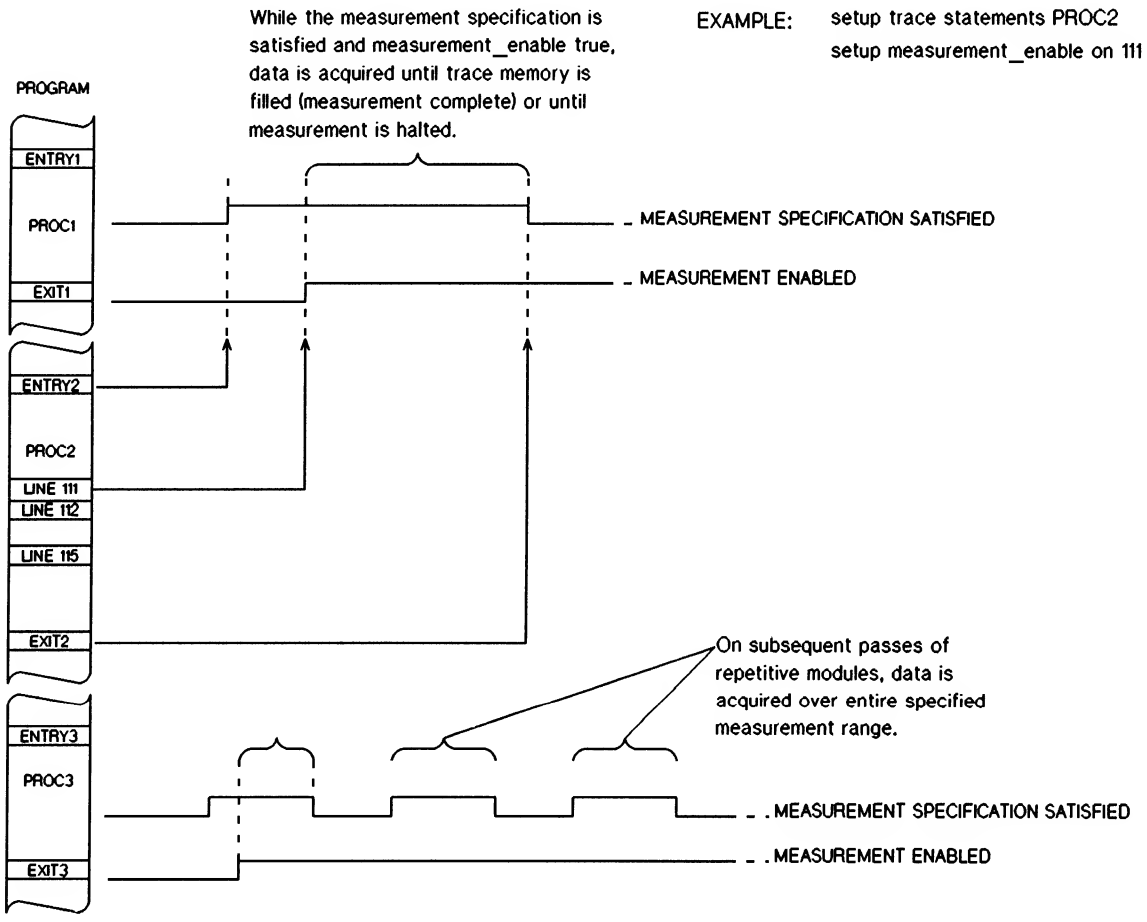


Figure 6-1. Measurement Enable

Measurement Disable

The ***measurement_disable*** command allows you to stop a measurement on execution of a specific event or sequence of events. This lets you halt program execution and view program activity leading up to the measurement disable term. The measurement disable function is illustrated in figure 6-2. When a measurement disable term is found, data acquisition is halted and the measurement is completed. If the trace memory is filled before the disable condition is found, data acquisition stops and the measurement is completed. Both conditions cause measurement completion as defined for the ***setup break*** and ***wait*** commands. The measurement disable function can be used in both ***real-time optional*** mode and ***real-time required*** mode. In both modes, the disable term will be displayed on the measurement display if it is part of the program segment being traced, e.g., ***setup trace modules*** PROC1, ***setup measurement_disable on*** PROC1 ***exit***. Sequential disable terms are not allowed in ***real_time optional*** mode.

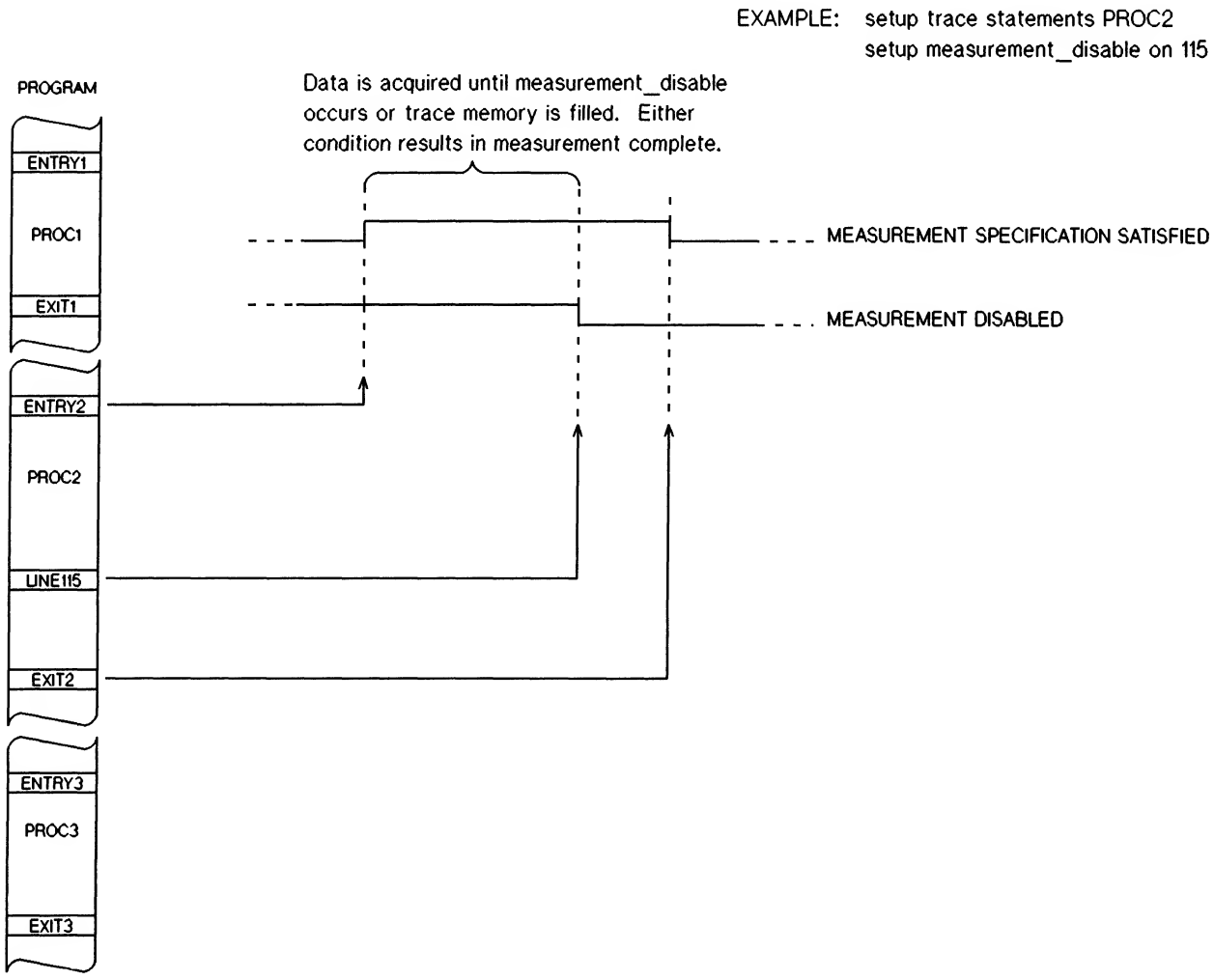


Figure 6-2. Measurement Disable

Windowing

The *setup measurement_disable window* command allows you to re-enable a measurement on an measurement enable term. Figure 6-3 shows the window function. When the measurement enable term occurs, the measurement is enabled. Measurement data is acquired while the measurement specification is satisfied. When the measurement disable term occurs, the measurement is disabled, and the analyzer searches for the next occurrence of the enable term. When the enable term is found, the analyzer acquires more data until disabled again. In this manner, repetitive measurements can be made on a code segment defined in the window. Windowing can only be used in *real-time optional* mode.

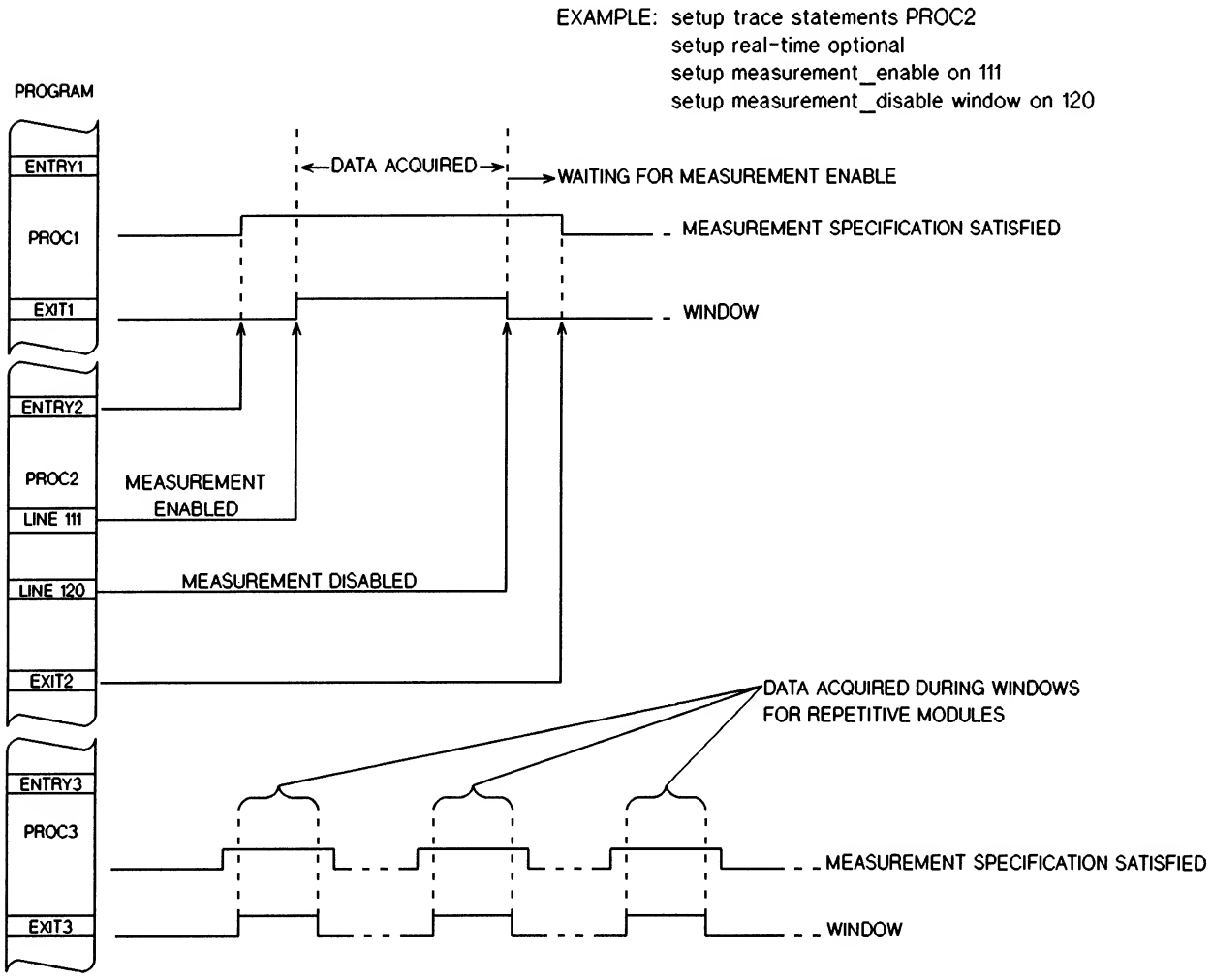


Figure 6-3. Windowing

Using Sequential Measurement Enable/Disable Terms

Sequential measurement enable and disable terms can be defined using the ***followed_by*** parameter. Sequential enable and disable terms enable you to uniquely define a software path in your code as the measurement enable or disable term. The terms may be source program lines or the entry and/or exit states of program modules. The functional operation of sequential terms is illustrated in figure 6-4.

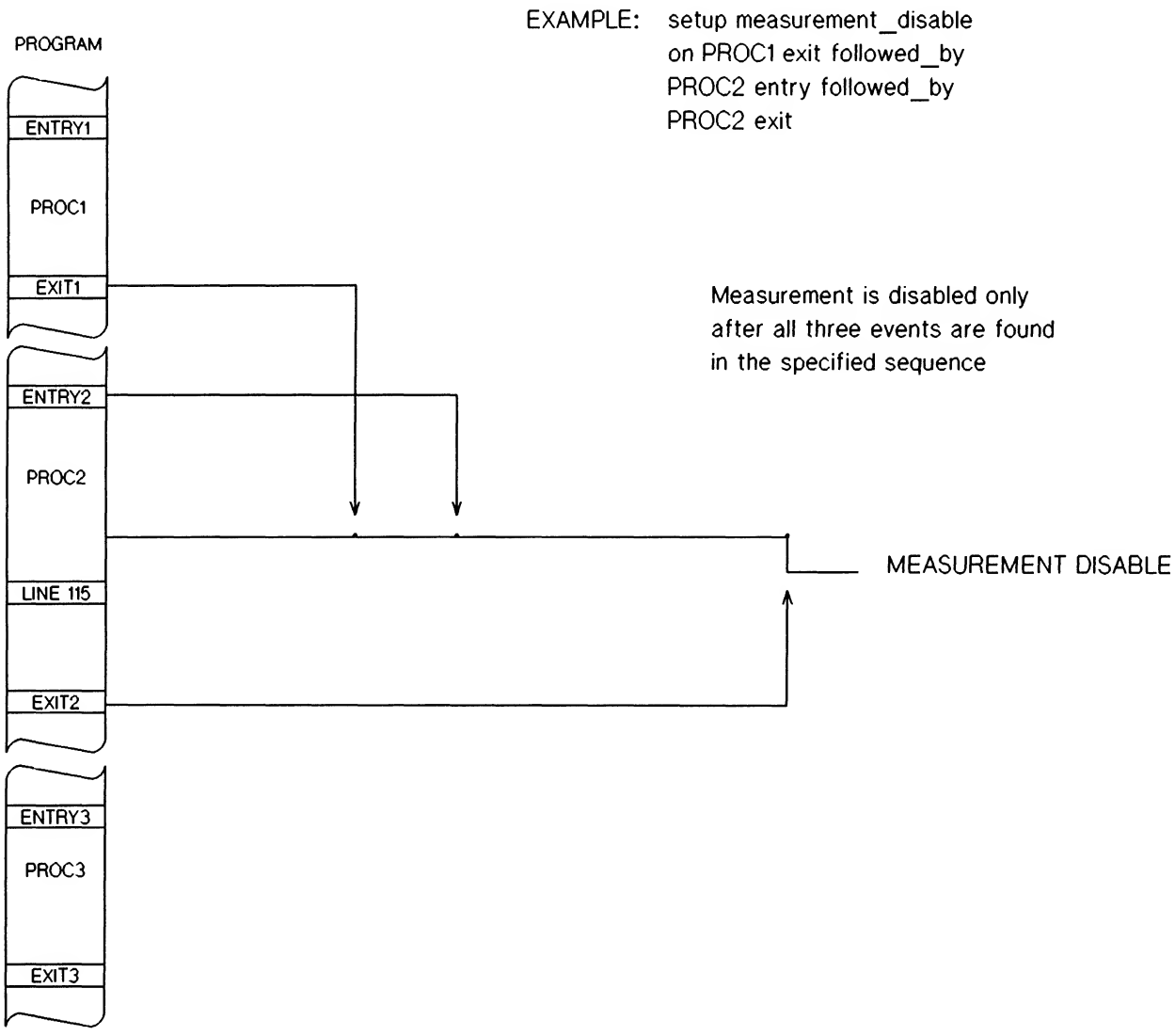


Figure 6-4. Using Sequential Enable/Disable Terms

Using OR'ed Measurement Enable/Disable Terms

You can specify measurement enable/disable terms as OR'ed combinations of terms. When OR'ed terms are used, the measurement is enabled or disabled on the first occurrence of any one of the OR'ed terms. The use of OR'ed terms is shown in figure 6-5.

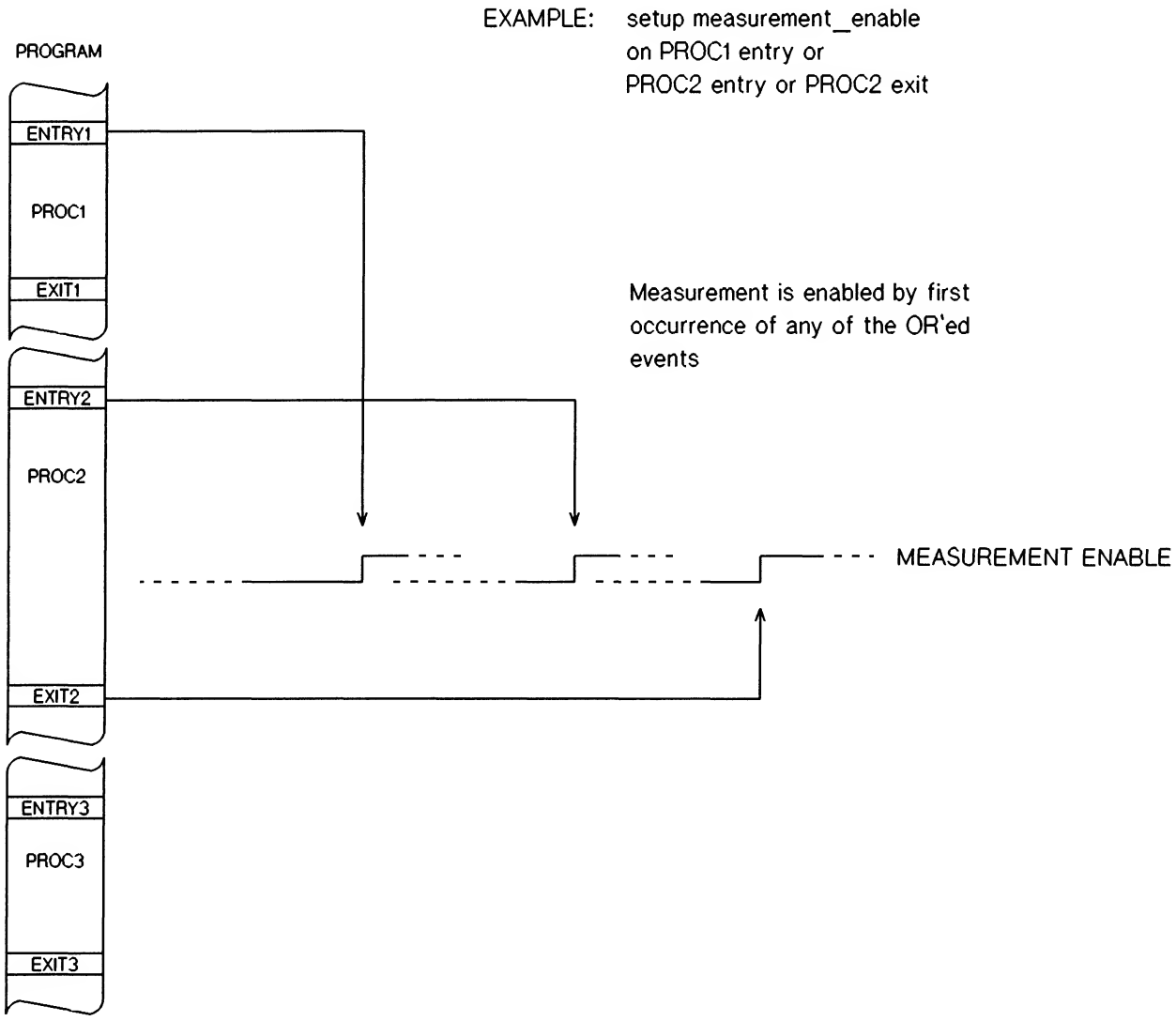


Figure 6-5. Using OR'ed Enable/Disable Terms

Number of Enable/Disable Terms

A combined total of up to six enable and disable terms can be specified in ***real-time required*** mode. These six terms can be used in any combination of OR'ed or sequential terms. In ***real-time optional*** mode, six enable terms are available and six or less disable terms are available. The number of disable terms available will vary, depending upon the measurement specified.

Interaction Between Measurement Enable/Disable and IMB

The software analyzer can interact with other HP 64000 system modules via the intermodule bus (IMB). This interaction is defined with the ***setup trigger_enable*** command, and the ***setup measurement_enable*** and ***setup measurement_disable*** commands. A measurement enable or disable condition must be defined in order to make interactive measurements over the IMB. If the enable or disable term is set to ***any_state***, the IMB specification (***setup trigger_enable***) controls the measurement enable or disable function of the software analyzer. If an enable or disable term is defined, that term is combined with the ***setup trigger_enable*** condition to define a sequential enable or disable condition. The ***any_state*** parameter should be used only when making interactive measurements over the IMB. When ***any_state*** is specified, one state must occur before the measurement is enabled. When operating your software analyzer stand-alone, this may cause data to be lost at the beginning of your measurement.

See chapter 11 for detailed information on IMB measurements.

TRIGGER ENABLE RECEIVED. If ***trigger_enable received*** is specified, a trigger enable must be received from another HP 64000 analysis subsystem before the software analyzer starts looking for the measurement enable or disable condition. The trigger enable becomes the first term in a sequential measurement enable or disable condition.

TRIGGER ENABLE DRIVEN. If ***trigger_enable driven_only*** is specified, the software analyzer first looks for its measurement enable or disable condition. Upon finding the measurement enable or disable condition, the software analyzer drives the trigger enable line high, enabling another HP 64000 analysis subsystem, if one is set up to receive trigger enable.

measurement_enable

measurement_enable is used with the ***setup*** command to define conditions that must be met to enable execution of the specified software analyzer measurement.

Syntax

The command syntax for the ***setup measurement_enable*** command is shown in figure 6-6.

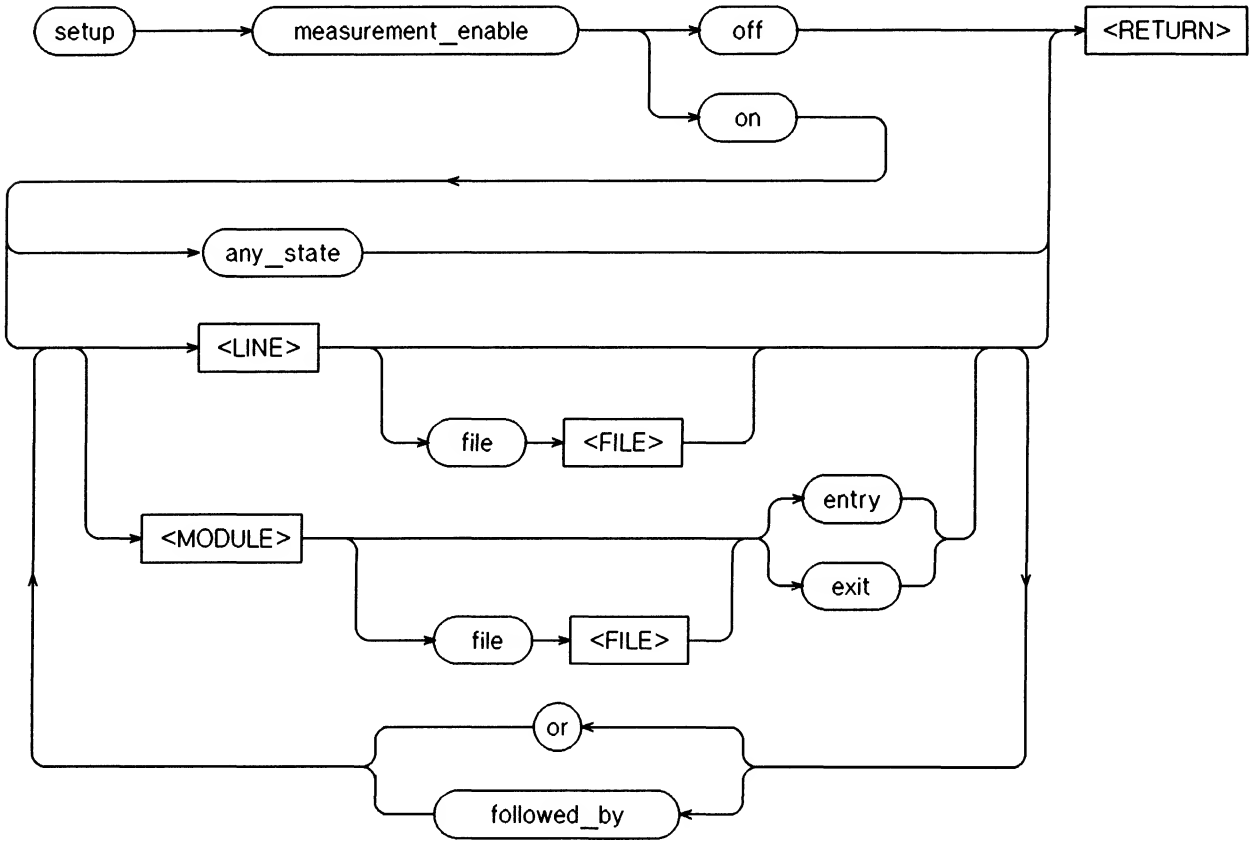


Figure 6-6. Setup Measurement_Enable Command Syntax

Default Value

off

measurement_enable (Cont'd)

Parameters

any_state — *any_state* specifies that measurement enable is to be set true on any state occurring in program execution. This is intended for use with IMB measurements. The *off* parameter should be used with stand-alone software analyzer measurements.

entry — *entry* specifies that the measurement be enabled on entry to the specified module.

exit — *exit* specifies that the measurement be enabled on exit from the specified module.

file — *file* indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the *file* softkey. If the module or line is in the defined default path, *file* may be omitted in the command statement.

<FILE> — <FILE> is an optional parameter that refers to the source file containing the line or module specified in the command statement. If the module or line is in the defined default path, the <FILE> parameter may be omitted in the command statement.

followed_by — *followed_by* is used to specify sequential enable conditions.

<LINE> — <LINE> represents the line number of a Pascal or C statement in the source program. If the specified <LINE> contains only comments (no executable code), the analyzer will associate the line number with the first line containing executable code following it.

<MODULE> — <MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure, function or the main program within a specified file. In C, a module can be the name of a function within a specified file.

off — *off* turns off the measurement enable function. This causes the analyzer to always be enabled.

on — *on* is a delimiter that indicates the measurement enable conditions immediately follow on the command line.

or — *or* is a logical operator for inclusive ORing of terms for the measurement enable conditions.

Examples

```
setup measurement_enable on 111 file BUB_SORT
setup measurement_enable on PROC2:BUB_SORT exit or 115
   followed_by PROC3 entry
setup measurement_enable any_state
setup measurement_enable off
```

measurement_disable

measurement_disable is used with the *setup* command to define conditions that must be met to disable execution of the specified software analyzer measurement.

Syntax

The command syntax for the *setup measurement_disable* command is shown in figure 6-7.

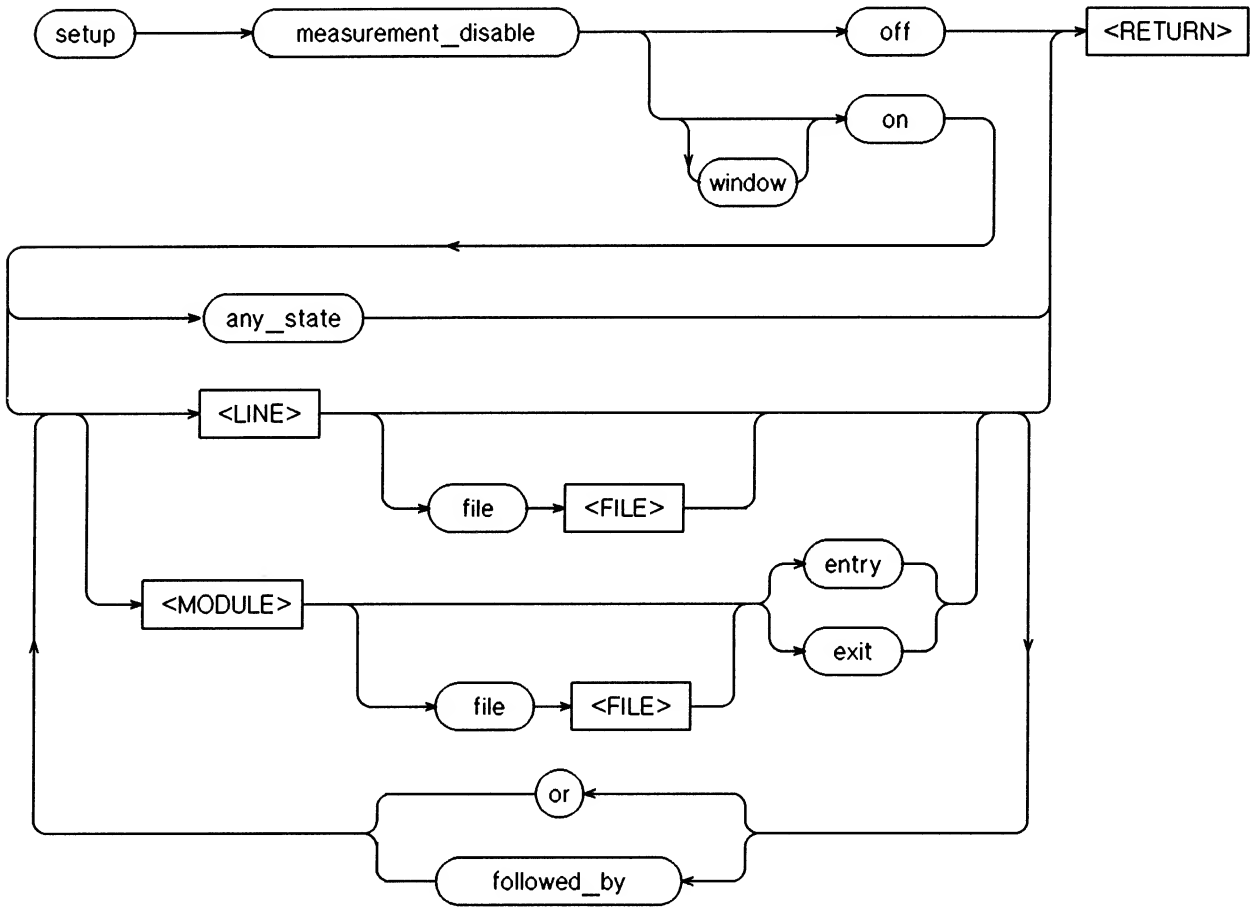


Figure 6-7. Setup Measurement_Disable Command Syntax

Default Value

off

measurement_disable (Cont'd)

Parameters

any_state — ***any_state*** specifies that measurement disable is to be set true on any state occurring in program execution. This is intended for use with IMB measurements. The ***off*** parameter should be used with stand-alone software analyzer measurements.

entry — ***entry*** specifies that the measurement be disabled on entry to the specified module.

exit — ***exit*** specifies that the measurement be disabled on exit from the specified module.

file — ***file*** indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the ***file*** softkey. If the module or line is in the defined default path, ***file*** may be omitted in the command statement.

<FILE> — <FILE> is an optional parameter that refers to the source file containing the line or module specified in the command statement. If the module or line is in the defined default path, the <FILE> parameter may be omitted in the command statement.

followed_by — ***followed_by*** is used to specify sequential disable conditions.

<LINE> — <LINE> represents the line number of a Pascal or C statement in the source program. If the specified <LINE> contains only comments (no executable code), the analyzer will associate the line number with the first line containing executable code following it.

<MODULE> — <MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure, function or the main program within a specified file. In C, a module can be the name of a function within a specified file.

off — ***off*** turns off the measurement disable function.

on — ***on*** is a delimiter that indicates the measurement disable conditions immediately follow on the command line.

or — ***or*** is a logical combinatoric for inclusive ORing of terms for the measurement disable conditions.

window — ***window*** allows the software analyzer to re-enable on the measurement enable condition after the measurement disable condition has been met. ***window*** allows you to make repetitive measurements in the program execution range specified by the "window".

measurement_disable
(Cont'd)

Examples

setup measurement_disable on PROC2:BUB_SORT *exit followed_by* PROC3 *entry*
setup measurement_disable window on 115
setup measurement_disable any_state
setup measurement_disable off

MEASUREMENT QUALIFICATION EXAMPLE

The following measurement example shows how the measurement enable and disable conditionals can be used to qualify the data acquired by the software analyzer.

Source Program Lines

The following source program segment is used in the measurement example:

```
 80    BEGIN  (*PROC2 MAIN BODY*)
 81
 82      PTR^.I := PTR^.I-1;
 83      Y:=1;
 84      D:=D-1; (*Scoped variable*)
 .
 .
 .

 92      COLOR_SET:= [ BLACK,BROWN ];
 93
 94      IF COUNT = 10
 95        THEN  COUNT:=0
 96        ELSE
 97          BEGIN
 98            COUNT := COUNT+1;
 99            COLOR_SET := COLOR_SET + [ WHITE, GREEN ];
100            PROC2(SNN,SNV,SVN,SVV,DN,DV,PTR);
101          END;
102
103      T:=P2;
104      S:=T*P2;
105      NEW_VALUE :=T-S;
106
107      IF P2 <> 0 THEN Y:=Y/P2;
108      IF NEW_VALUE <> 0 THEN  TIME_VARIABLE := T*((S+Y)/NEW_VALUE);
109      U.FLAG := TRUE;
110      IF U.FLAG = TRUE THEN U.N:=T+S;
111
112   END;   (*PROC2 MAIN BODY*)
```

Measurement Setup

For this measurement example, the software analyzer is setup to repetitive trace execution of the IF .. THEN .. ELSE statement on source program lines 94 through 101. This is done by tracing statements in procedure PROC2 with a window defined around the IF .. THEN .. ELSE statement. This is accomplished with the following series of commands:

setup real_time optional
setup measurement_enable on 94
setup measurement_disable window on 100
setup trace statements PROC2

Note that ***real_time optional*** is required when using the window function. The ***file*** option is omitted in the command statements because PROC2 is contained in the default path. See the setup display in figure 6-7.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

TRACE STATEMENTS

     module / line          file
     PROC2                  NT1:TESTP

ENABLE on
     line 94 file NT1:TESTP

DISABLE window on
     line 100 file NT1:TESTP

RUN_AT_EXECUTION from
     transfer_address

DEFAULT_PATH
     file NT1:TESTP

STATUS: Database search successful                                    16:19

 setup  trace  statements  PROC2


  run    setup  db check  display  modify   show   execute  ---ETC---
```

Figure 6-7. Setup Display For Trace Qualification Example

Measurement Display

Executing the measurement results in the measurement display shown in figure 6-8. Note that the analyzer has repetitive traced lines 94,98,99, and 100, program execution of the ELSE statement in the program listing. This is what we expect to see, since the value of COUNT is not equal to 10. (See Symbol and Value fields opposite line 99. The Symbol and Value fields are offset from their corresponding statements because of processor instruction prefetch.) By rolling the measurement display up until we see a COUNT value of 10 (figure 6-9), we see the THEN statement being executed at line 95. Since the disable condition is in the ELSE statement which is not executed, the analyzer continues to trace statements after line 100.

This example illustrates how measurement qualification can be used to trace program execution at precisely the location you need to look at, simplifying your software analysis task.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Source                                  Symbol      Value       Stat   Time-rel
Break for new stack information
  94 IF COUNT = 10                                                      0.0 uS
  98 COUNT := COUNT+1;                                                  1.3 uS
  99 COLOR_SET := COLOR_SET + [ WHIT* COUNT                 0 read      1.2 uS
  99      "                           COUNT                 1 write     2.6 uS
 100 PROC2(SNN,SNV,SVN,SVV,DN,DV,PTR* COLOR_SET    [BLACK,BROW* read      2.3 uS
Break for new stack information
                        -- Window disable occurred --
Break for new stack information
  94 IF COUNT = 10                                                      0.0 uS
  98 COUNT := COUNT+1;                                                  1.5 uS
  99 COLOR_SET := COLOR_SET + [ WHIT* COUNT                 1 read      1.2 uS
  99      "                           COUNT                 2 write     2.4 uS
 100 PROC2(SNN,SNV,SVN,SVV,DN,DV,PTR* COLOR_SET    [BLACK,BROW* read      2.1 uS
Break for new stack information

STATUS: Awaiting Command                                        30    16:12

   run    setup  db check  display  modify   show    execute  ---ETC---
```

Figure 6-8. Measurement Display Showing ELSE Statement Execution

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Source                                  Symbol      Value       Stat   Time-rel
Break for new stack information
  94 IF COUNT = 10                                                      0.0 uS
  98 COUNT := COUNT+1;                                                  1.3 uS
  99 COLOR_SET := COLOR_SET + [ WHIT* COUNT                 0 read      1.2 uS
  99      "                           COUNT                 1 write     2.6 uS
 100 PROC2(SNN,SNV,SVN,SVV,DN,DV,PTR* COLOR_SET    [BLACK,BROW* read      2.3 uS
Break for new stack information
                        -- Window disable occurred --
Break for new stack information
  94 IF COUNT = 10                                                      0.0 uS
  98 COUNT := COUNT+1;                                                  1.5 uS
  99 COLOR_SET := COLOR_SET + [ WHIT* COUNT                 1 read      1.2 uS
  99      "                           COUNT                 2 write     2.4 uS
 100 PROC2(SNN,SNV,SVN,SVV,DN,DV,PTR* COLOR_SET    [BLACK,BROW* read      2.1 uS
Break for new stack information

STATUS: Awaiting Command                                        30    16:12

   run    setup  db check  display  modify   show    execute  ---ETC---
```

Figure 6-9. Measurement Display Showing THEN Statement Execution

NOTES

Chapter 7

CONTROLLING THE EMULATOR

OVERVIEW

This chapter provides the following information:

- A description of the software analyzer/emulator interface
- An explanation of how the analyzer and emulator communicate with each other.
- How to use the emulation monitor
- Detailed descriptions of the emulation commands executable from the software analyzer.

GENERAL INFORMATION

The software analyzer uses the emulation subsystem or your target system as an execution environment. The software analyzer includes a subset of the emulator commands to enable you to control emulation from within the analyzer. These commands are ***break, load, reset,*** and ***run.*** These are the four basic commands needed to control a user's program running in emulation memory. The incorporation of the emulator commands simplify the interface between you and the system by providing the means for you to control the emulator without exiting the software analyzer.

EMULATION INTERFACE

Emulation Configuration File

When you invoke the software analyzer with the *sw_anl_N* command, you must specify a file name. This file name can be the name of a emulation command file or the name of a software analyzer configuration file. When both file types exist with the same file name, the software analyzer configuration file is used.

The emulation command file is the command file that was used in emulation to configure the emulator for a particular application. This file is generated during the emulation session and contains your answers to a series of questions ending with "*Command file name?*". This command file name is used to create a file of type "emul_com" (emulation command). The software analyzer uses this command file to determine which emulator is used for software analysis. This configuration file is also used to determine the state of the emulator. When an emulation command file is

specified, the software analyzer is always reconfigured to the specified emulator. The current software analyzer configuration is lost.

The software analyzer configuration file is created with the *configuration save_in* command during a software analysis session. The software analyzer uses the configuration file to determine which emulator is used for software analysis. The configuration file also configures the software analyzer for a measurement.

When no file is specified, the software analyzer prompts you with the question "*Emulation command file?*". You must specify an emulation command file before the system will allow you access to the software analyzer.

Loading The User Program

User programs can be loaded from within the software analyzer module (refer to the descriptions of the emulation commands and their syntax in this chapter). After the emulation configuration is complete, load the absolute file into emulation memory using the *load* command. If the address range into which a program is to be loaded resides entirely in internal emulation memory, the processor remains in the reset state. If any portion of the program resides in memory which has been mapped as external user memory, the processor is released from the reset state. After loading all portions of the file which are to reside in emulation memory, a handshake is performed to determine if the processor is executing in the emulation monitor program and, if not, a break is performed. When the processor is in the monitor, the user memory portion of the program is loaded. This sequence can be performed manually by using the options of the *load* command which specify the portion of memory to be loaded.

If your program resides in ROM in your target system, the absolute file name must be specified with the *setup absolute_file* command in order for the software analyzer to perform measurements on the code.

NOTE

When using the HP 64243AA/B or HP 64245AA/AB emulators, the absolute file will be loaded to the last address space specified in the emulator, i.e. supervisor or user program space, supervisor or user data space, etc.

NOTE

When the emulator is running in the monitor, the processor must be reset before an absolute file containing the emulation monitor program can be loaded.

Selecting The Emulation Analysis Mode (64243,64245 Emulators only)

The emulator analysis mode must be set to *bus_cycle_data* in order to use the software analyzer. From within the emulation subsystem, execute the command *modify analysis_mode_to bus_cycle_data*. If the emulation analysis mode is not set to *bus_cycle_data*, the error message "**Incorrect analysis bus mode for this analyzer**" is displayed on the status line when you attempt to access the software analyzer.

Running The User Program

Once the program has been loaded, release the processor from the reset state with the ***run*** command. If the command is issued as the single keyword ***run***, the processor will use the start-up vector or routine to start execution in the emulation monitor. The status line will display "***Running in monitor***" indicating that the HP 64000/monitor handshake is being performed.

RUNNING YOUR PROGRAM IN REAL-TIME OPTIONAL MODE. From the emulation monitor, you can use the single keyword ***run*** to start execution of the your program in ***real_time optional*** mode. When the command ***run*** is given, program execution begins at the transfer address specified in the source program. Thereafter, ***run*** will cause execution to begin at the address contained in the program counter (PC) register.

RUNNING YOUR PROGRAM IN REAL-TIME REQUIRED MODE. In ***real_time required*** mode, executing the single keyword ***run*** will release the processor reset line. To start execution of the user program, you must specify a ***from*** state in the ***run*** command statement.

COMMUNICATION BETWEEN THE SOFTWARE ANALYZER AND EMULATION

The software analyzer communicates with the emulation processor by transferring data to and from emulation memory. Data transfer is accomplished through the memory controller board into the emulation memory boards. The memory controller contains a hardware mapper that is programmed by the emulation command file to map the emulation processor address space into emulation or user memory spaces designated as RAM and/or ROM memory.

The software analyzer controls the emulation processor reset and break functions directly through the emulation control board. Refer to your HP 64000 System Emulation/Analysis manual for a more detailed description of how the HP 64000 host processor controls emulation.

The software analyzer and the emulator communicate the status of the emulator hardware to each other. Whenever the emulation hardware is modified by either the software analyzer or the emulator, the hardware change is reflected when the other module is entered. Note that the status of the hardware is communicated to the other module only if the modules are exited using the ***end*** softkey.

USING THE EMULATION MONITOR

The software analyzer makes extensive use of the emulation monitor in the ***real_time optional*** mode. If ***real_time optional*** is specified, the emulation monitor must be linked with your program and must reside in emulation memory. The emulation monitor is not required in the ***real_time required*** mode. The monitor supplied with the emulation software is designed to work with the software analyzer. If you have modified this monitor, it is possible that the software analyzer may not function properly. To verify that a modified monitor is functioning properly, perform the following procedure:

1. Access the emulator and load the absolute file containing the modified monitor.

2. Set the emulator such that it is running in the emulation monitor.

3. Verify that the ***display registers***, ***modify memory***, and ***display memory*** commands execute correctly.

4. Verify that program execution is transferred to the monitor and that the "***running in monitor***" message is displayed when a break is executed.

All of the preceding features and functions must execute correctly to ensure proper operation of the software analyzer. If any of the above steps fail, modify your emulation monitor until the problem is corrected.

break

The *break* command causes the processor to be diverted from execution of the user program to the emulation monitor. A break is defined as a transition from execution of a user's program to the Emulation Monitor.

Syntax

The syntax for entering the *break* command is shown in figure 7-1.

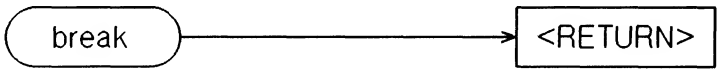


Figure 7-1. Break Command Syntax Diagram

Default Value

None

Parameters

None

Break Command Example

break

load

The *load* command transfers absolute code from the HP 64000 system disc into user RAM or emulation memory. The destination of the absolute code is determined by the memory configuration map which was set up during emulation configuration and the address specified during linking. When using the HP 64243AA/B or HP 64245AA/AB emulators, the absolute file will be loaded to the last address space specified in the emulator, i.e., supervisor or user program space, supervisor or user data space, etc.

NOTE

When the emulator is running in the monitor program and a *load* command is given which reloads the monitor, the results are unpredictable. If a reload of the monitor is required, first put the emulator in the reset mode.

Syntax

The *load* command syntax is shown in figure 7-2.

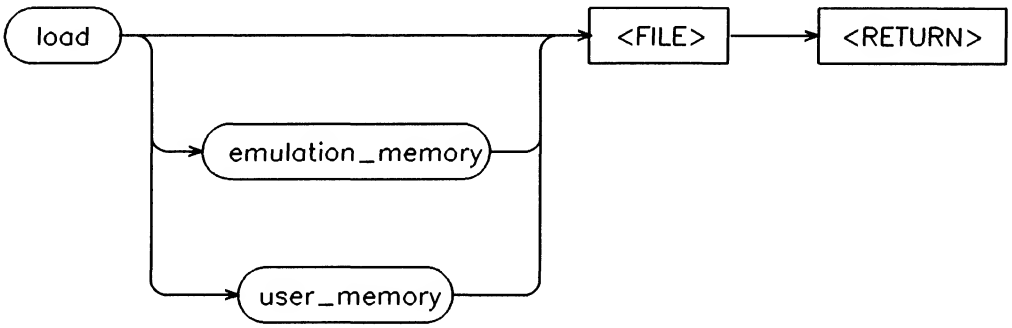


Figure 7-2. Load Command Syntax Diagram

Default Value

all memory

Parameters

emulation_memory — *emulation_memory* specifies that absolute code is to be loaded into emulation memory. The destination of the absolute code is determined by the address specified during linking.

load (Cont'd)

<FILE> — <FILE> is the identifier of the absolute file to be loaded from the HP 64000 system memory into user RAM or emulation memory The syntax requirements for <FILE> are discussed in Appendix B.

user_memory — *user_memory* specifies that the absolute program be loaded into user RAM in the target system. In the context of the load command, *user_memory* refers to target system memory.

Load Command Examples

load TESTP
load emulation_memory TESTP
load user_memory TESTP

reset

The *reset* command suspends target system operation and reestablishes initial operating parameters, such as reloading control registers. The reset signal is latched when active and is released by the *run* command.

Syntax

The command syntax for executing the *reset* command is shown in figure 7-3.

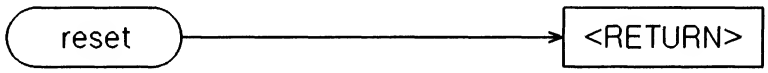


Figure 7-3. Reset Command Syntax Diagram

Default Value

None

Parameters

None

Reset Command Example

reset

run

When the processor is in a reset state, ***run*** causes the reset to be released, and if a ***from*** address is specified, the processor is directed to that address. The program can either be run from (1) the transfer address of the user's program, (2) a specified address, (3) a specified line number in the source code, (4) from the entry point of a specified module, (5) from the address currently stored in the processor's program counter, or (6) from a global symbol.

When the single keyword ***run*** is executed when the processor is reset, the reset vector directs program execution to the emulation monitor. When ***run*** is executed after a break, program execution begins with the next PC.

RUNNING IN REAL-TIME OPTIONAL MODE.
In ***real_time optional*** mode, executing the single keyword ***run*** while in the emulation monitor causes the user program to start executing from the transfer address specified in the source program.

RUNNING IN REAL-TIME REQUIRED MODE.
In ***real_time required*** mode, executing the single keyword ***run*** while in the emulation monitor simply restarts the emulation monitor through the reset vector. A ***from*** term must be specified to begin execution of the user program when in the emulation monitor with real-time required.

RUN AT_EXECUTION. A "***run at_execution***" command causes the user's program to start running after the ***execute*** softkey has been pressed. This enables the trace measurement to be started before beginning program execution, ensuring that the analyzer can trace all code executed starting with the "***run from***" location.

NOTE

When using the ***run at_execution*** command, do not configure your measurement for ***measurement_enable on any_state*** (unless used with ***trigger_enable***) or ***trace statements*** **"don't care"**. Either of these configurations may cause the analyzer to acquire invalid data (the emulation monitor).

Syntax

The syntax for executing the ***run*** command is shown in figure 7-4.

Default Value

If no ***from*** option is specified with the ***run*** command, the emulator will begin program execution at the current address specified by the processor's program counter.

run
(Cont'd)

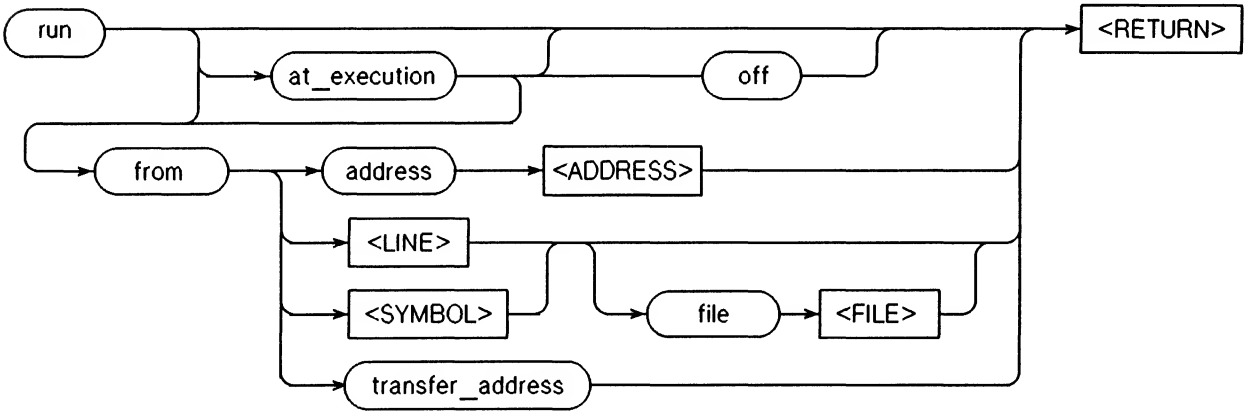


Figure 7-4. Run Command Syntax Diagram

Parameters

address — ***address*** indicates the information that follows is an address constant specified in binary, octal, decimal, or hexadecimal.

<ADDRESS> — <ADDRESS> represents an address within the absolute file loaded into user or emulation memory from which the processor will begin program execution. The syntax allows specification of a positive or negative offset from the absolute address.

at_execution — ***at_execution*** causes the program to start running from a specified location at execution of a trace measurement.

file — ***file*** indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the ***file*** softkey.

<FILE> — <FILE> represents the name of the source file containing the address, line, or symbol from which the processor is to begin program execution.

from — ***from*** specifies that the location in the user's program from which program execution will begin follows in the command line.

<LINE> — <LINE> allows you to specify a line number in the source code as the starting point for program execution. Program execution begins at the absolute address containing the first executable instruction associated with the source line.

off — ***off*** turns off the ***at_execution*** parameter.

run (Cont'd)

<SYMBOL> — <SYMBOL> allows you to specify program execution to run from a specified symbol. If a file name is specified with <SYMBOL>, the analyzer assumes that the symbol is a module in the specified file. If no file is specified with <SYMBOL>, the analyzer first looks for the address of a global symbol in the link_sym file associated with the currently loaded absolute file. If no global symbol is found there, the analyzer then searches for a module in the current default file.

transfer_address — *transfer_address* specifies that the emulator begin program execution at the address stored in the transfer buffer (XFR_BUF). This is the starting address of the user program.

Run Command Examples

run
run at_execution from transfer_address
run from 173 *file* TESTP
run from PROC2
run from address 3490H

NOTES

Chapter 8

MAKING TRACE MEASUREMENTS

OVERVIEW

This chapter describes the following software analyzer trace measurements:

- Trace data_flow
- Trace modules
- Trace statements
- Trace variables

GENERAL INFORMATION

The software analyzer has four trace measurement modes. These modes are (1) trace data_flow , (2) trace modules, (3) trace statements, and (4) trace variables. This chapter provides detailed descriptions of each trace measurement mode, including a general description of the measurement, any anomalies that may exist in the measurement, syntax diagrams, softkeys used in each operating mode, and examples of each measurement.

If you have any difficulties or problems when executing trace measurements, see appendix E, Resolving Measurement Problems, for possible solutions.

NOTE

The software analyzer does not distinquish between supervisor and user modes. If these modes are used to map multiple physical addresses to one logical address, the software analyzer will correlate all physical addresses with the last program loaded. This will probably result in erroneous data being displayed in the measurement display.

trace data_flow

The trace data_flow measurement traces the values of specified variables or parameters on entry to and exit from selected procedures or functions in a program. The traced variables must be accessible at the procedure entry point, exit point, or both. Static variables can always be accessed. Local variables can be accessed only if the variable (1) belongs to a parent procedure, (2) is a pass-by-reference parameter to the specified procedure, or (3) is a pass-by-value parameter and measurement is with procedure entry.

If the variable is local to the associated module, it can never be accessed since none of a module's local variables are created until after module entry and they are removed from the stack before module exit. Value parameters are active only at procedure entries, and reference parameters are always active with respect to their procedure. If a value parameter is requested on exit to its procedure, a warning message will be displayed and no values of that parameter will be displayed.

Up to ten symbols may be specified in the ***setup trace data_flow*** command in combinations of procedures (functions) and variables. A maximum of three procedures can be traced. For example, the setup command could call for nine variables to be traced in one procedure, four variables to be traced in each of two procedures, or a total of seven variables to be traced in three procedures.

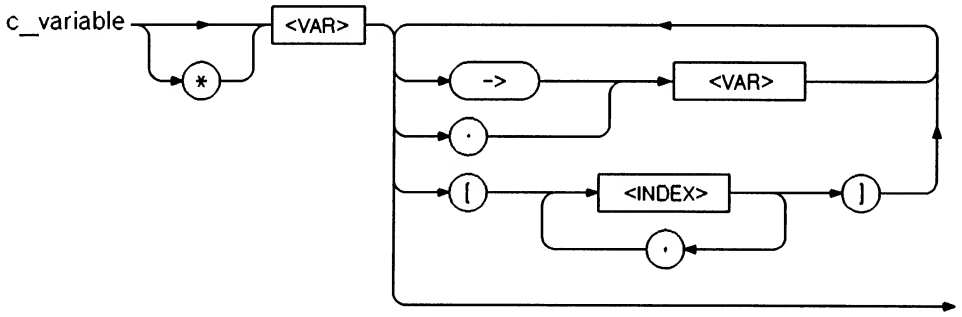
Command Syntax

The command syntax for setting up the trace data_flow measurement is shown in figure 8-1.

Parameters

The following paragraphs define the parameters used in the ***setup trace data_flow*** command.

c_variable — c_variable may be any valid C variable in the of the following expression format.



entry — ***entry*** specifies that data be traced only on entry to the specified module(s). The default value is to trace data on both entry to and exit from a module.

exit — ***exit*** specifies that data be traced only on exit from the specified module(s).

trace data_flow (Cont'd)

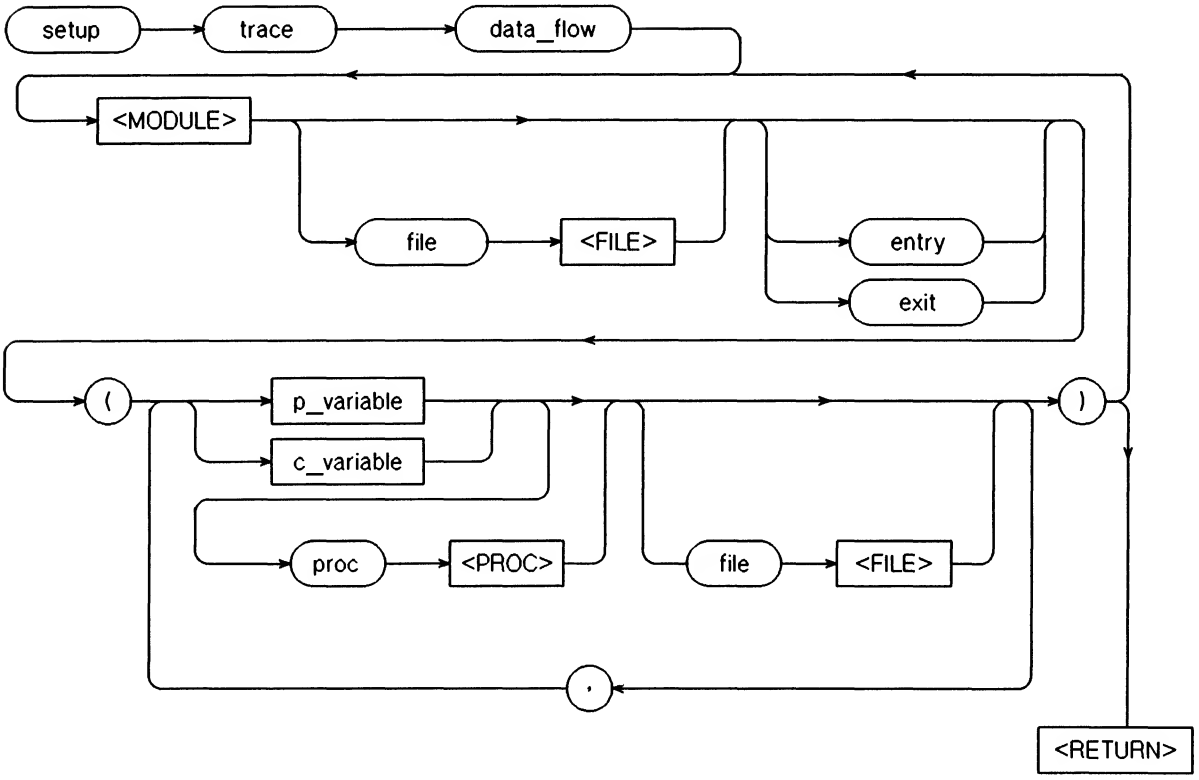


Figure 8-1. Setup Trace Data_Flow Syntax Diagram

file — *file* indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the *file* softkey.

<FILE> — <FILE> is an optional parameter that refers to the source file containing the specified <MODULE>, <VAR>, or <PROC> called out in the command statement. If the <MODULE>, <VAR>, or <PROC> is in the defined default path, the <FILE> parameter may be omitted from the command statement.

<INDEX> — Represents an index value (integer or scalar value) specifying a component of an array.

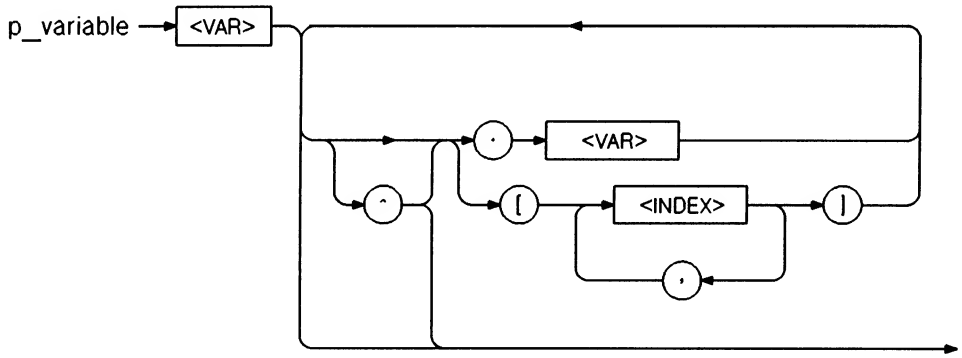
<MODULE> — <MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure or a function within a specified file. In C, a module can be the name of a function within a specified file. The trace data_flow measurement traces the specified variables on entry to and/or exit from the <MODULE> as specified in the command line. A given module can only be specified once.

trace data_flow (Cont'd)

proc — ***proc*** indicates that a procedure or function name follows that defines the procedure or function to which a variable belongs. NOTE: an "@" may be used in place of ***proc***.

<PROC> — <PROC> is an optional parameter that refers to a procedure or function in which <VAR> is declared. If <PROC> is defined in the ***setup default_path*** command, it may be omitted in the ***setup trace data_flow*** command. If <PROC> is not specified in either the default path or the ***setup trace data_flow*** command, the analyzer assumes that <VAR> is a global variable defined at the main program level.

p_variable — p_variable may be any valid Pascal variable in the following expression format.



<VAR> — <VAR> represents the name of a variable or parameter to be traced on entry to and/or exit from a <MODULE>. <VAR> can be any valid Pascal or C variable expression.

Setup Command Examples

The following command examples illustrate how to use the ***setup trace data_flow*** command to define measurements.

setup trace data_flow PROC2(COUNT,D ***proc*** PROC1,PTR ***proc*** PROC2)
setup trace data_flow PROC1 (SN ***proc PROC1 , SV proc*** PROC1)
PROC2 (SNN ***proc*** PROC2 , SVN ***proc*** PROC2 , COUNT)
setup trace data_flow PROC1 (AR[1,2,3] , RC.E1.EZ)
setup trace data_flow PROC1 (A^.B^.C^)
setup trace data_flow proc2 (*a->b->c)

Trace Data_Flow Measurement Example

The following example shows several lines of a program, a setup trace data_flow command for the program segment, and the resulting trace display.

trace data_flow (Cont'd)

SOURCE PROGRAM LINES. The following source program segment is traced in the *trace data_flow* measurement example.

```
137  PROCEDURE PROC4(A:INTEGER);
...
143  PROCEDURE PROC10(XV:INT; VAR XN:INT; YV:PTR; VAR YN:PTR);
...
151  BEGIN  (*MAIN PROGRAM*)
...
183    PROC4(COUNT+2);
...
201    PROC10(X,X,Y,Y);
...
```

SETUP MEASUREMENT COMMAND. The following *setup trace data_flow* command results in the setup display shown in figure 8-2. Note that since A[RED] and X are defined at the program level, they are not associated with a "proc" in the setup display.

setup trace data_flow PROC10 (XV *proc* PROC10, XN *proc* PROC10, YV *proc* PROC10, YN *proc* PROC10, A[RED]) PROC4 (X)

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

TRACE DATA_FLOW

    module     variable                proc          file
    PROC10               entry_exit                  NT1:TESTP
               XV                      PROC10        NT1:TESTP
               XN                      PROC10        NT1:TESTP
               YV                      PROC10        NT1:TESTP
               YN                      PROC10        NT1:TESTP
               A[RED]                                NT1:TESTP
    PROC4                entry_exit                 NT1:TESTP
               X                                     NT1:TESTP

ENABLE off

DISABLE off


STATUS: Database search successful ____________________________________ 16:19

setup trace data_flow PROC10  ( XV proc PROC10 , XN proc PROC10  , YV  proc PROC1
0 , YN proc PROC10 , A[RED] ) PROC4 ( X )

  run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 8-2. Trace Data_Flow Setup Display

trace data_flow (Cont'd)

MEASUREMENT DISPLAY. Figure 8-3 shows the measurement display resulting from the preceding setup specification. The trace list shows that PROC4 is called from line 183 and PROC10 is called from line 201. The values of the variables are shown immediately following the entry or exit of the corresponding module. Note that when PROC10 is exited, XV and YV are not active and are not displayed.

Note also that YV and YN are pointers and their values are the values of the pointers themselves, not the objects of the pointers. The software analyzer will trace the object of a pointer. If the variable YN^ (*yv in C) had been specified, the value of the object pointed to by pointer YV would have been displayed in the trace list. The software analyzer can trace seven levels of indirection.

In the C programming language, array parameters without an explicitly defined size cannot be traced as a whole.

Source lines displayed by trace data_flow measurements are not affected by instruction prefetch mechanisms. Source lines are not shown for exits.

```
64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Symbol              Value        Stat  Source
PROC4                            entry  183 PROC4(COUNT+2);
  X                      -1
PROC4                            exit
  X                      -1
PROC10                           entry  201 PROC10(X,X,Y,Y);
  XV                     10
  XN                     10
  YV              00000300CH
  YN              00000300CH
  A[RED]                 RED
PROC10                           exit
  XN                     11
  YN              00000300CH
  A[RED]                 RED
PROC4                            entry  183 PROC4(COUNT+2);

STATUS: Awaiting Command                                    20   16:12


  run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 8-3. Trace Data_Flow Measurement Display

trace modules

The trace modules measurement provides an overview of a program's control flow at the module level. This measurement allows you to isolate a problem to a specific module and provides a history of the module calls leading up to the problem. The entry and exit points of procedures and functions are traced and displayed with indentation used to indicate the level of nesting of the traced modules. The measurement can be set up to measure all modules or selected modules within a file or group of files. A total of 135 to 279 modules can be traced, depending on the number of files being traced.

When tracing recursive modules, each successive level of recursion is indented in the trace list. For large numbers of recursion levels, this may result in the data being shifted off the right side of the trace list display. This is indicated by an asterisk (*) displayed in the last display column. Recursive modules are indented relative to the outermost recursion level traced.

If a module name is longer than the symbol field width or the recursion level is deep, an asterisk is displayed in the last column of the symbol field. To display the entire name of a module, increase the symbol field width using the display command.

Command Syntax

The command syntax for setting up the ***trace modules*** measurement is shown in figure 8-4.

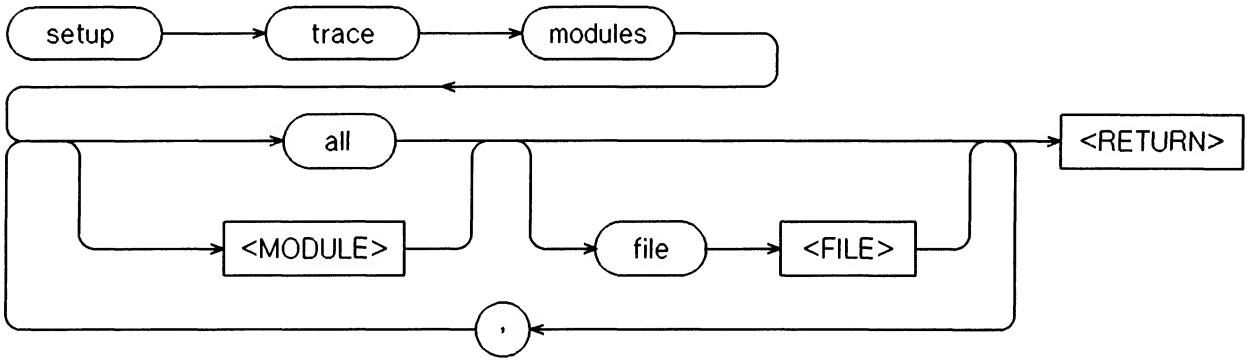


Figure 8-4. Setup Trace Modules Syntax Diagram

Parameters

The following definitions describe the parameters used in the ***setup trace modules*** command.

all — ***all*** specifies that all modules in the designated file or default path be traced. A maximum of 255 modules may be traced in one file.

trace modules (Cont'd)

file — *file* indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the *file* softkey.

<FILE> — <FILE> is an optional parameter that refers to the source file containing the specified modules called out in the command statement. If the <MODULE> is in the defined default path, the <FILE> parameter may be omitted from the command statement.

<MODULE> — <MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure or a function within a specified file. In C, a module can be the name of a function within a specified file. The trace modules measurement traces the entry and exit points of the specified modules.

Setup Command Examples

The following command examples illustrate how to use the *setup trace modules* command to define measurements.

```
setup trace modules all
setup trace modules all file BSORT , all file TESTP ,
    PROC1 , PROC2
setup trace modules PROC1 , PROC4 , PROC5
```

Trace Modules Measurement Example

The following example shows the use of the *setup trace modules* command and shows a sample setup display and the resulting measurement display.

SETUP MEASUREMENT COMMAND. The following *setup trace modules* command specifies that the software analyzer trace all modules in file Util and in file Fact.

```
setup trace modules all file Fact, all file Util
```

Figure 8-5 shows the setup display resulting from execution of the setup command.

trace modules (Cont'd)

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

TRACE MODULES

     module          file
     all             Fact:TESTP
     all             Util:TESTP

RUN_AT_EXECUTION from
     transfer_address

DEFAULT_PATH
     file TEST_68:TESTP

REAL_TIME
     optional



STATUS: Database search successful                                    16:19

 setup  trace  modules  all  file  Fact , all  file  Util


  run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 8-5. Trace Modules Setup Display

MEASUREMENT DISPLAY. The trace list in figure 8-6 shows the sequence of entries and exits for all modules in source files Util and Fact. In this example procedure **factorial** is recursive. The recursive descent can be seen in the succession of "entry"s in the Status field and also by the indentation of the procedure name in the Symbol field. The recursive ascent is shown in a similar manner.

Source lines displayed by trace modules measurements are not affected by instruction prefetch mechanisms. Source lines are not shown for exits.

trace modules
(Cont'd)

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Symbol                    Stat   Time-rel Source
 swap_elements_a          entry   12.6 uS  105 swap_elements_at (large_index,cu
 swap_elements_a          exit     5.3 uS
sort                      exit     5.2 uS
factorial                 entry  171.0 uS  133 factorial (value);
 factorial                entry  579.0 uS  121 else { descend_factor = value *
  factorial               entry  580.7 uS  121 else { descend_factor = value *
   factorial              entry  579.5 uS  121 else { descend_factor = value *
    factorial             entry  580.9 uS  121 else { descend_factor = value *
     factorial            entry  577.6 uS  121 else { descend_factor = value *
      factorial           entry  580.1 uS  121 else { descend_factor = value *
       factorial          entry  579.5 uS  121 else { descend_factor = value *
        factorial         entry  580.3 uS  121 else { descend_factor = value *
        factorial         exit    13.7 uS
       factorial          exit    13.6 uS
      factorial           exit    13.8 uS

STATUS: Awaiting Command                                          20    16:12


   run     setup   db check   display   modify    show    execute   ---ETC---
```

Figure 8-6. Trace Modules Measurement Display

trace statements

The trace statements measurement gives you a detailed view of a small section of code. The measurement traces the execution of source language statements and variables in each statement in a defined source program line range or a specified module. Each source statement is displayed with its line number and the value of variables referenced in the source statement. **NOTE**: the value of dynamic variables are not displayed in real-time required mode.

Some processors prefetch instructions prior to their execution. Prefetches have the following effects on the trace statements measurements:

1. Accesses to variables by instructions executed immediately prior to the address range being traced will appear as accesses occurring within the address range.

2. Accesses to variables by the last instructions executed within the address range being traced may not appear.

3. The symbol and value fields in the display may be offset from their corresponding source lines.

The trace statements measurement only displays variables accessed by the statements being traced. Any accesses caused by procedures or functions outside of the traced range are not shown. For example, if a variable is modified by a compiler library, that variable will not appear in the trace statements trace list.

A "don't care" trace statements measurement can be set up by entering a ***trace statements*** command with no parameters (***setup trace statements*** RETURN). When the measurement is enabled, the software will trace all bus states, including emulation monitor code and library routines. This measurement can be useful in determining which library files are called by a source statement and in determining where your program may have went "into the weeds". **NOTE: ALWAYS use this measurement in REAL_TIME REQUIRED mode.**

Command Syntax

The command syntax for setting up the trace statements measurement is shown in figure 8-7.

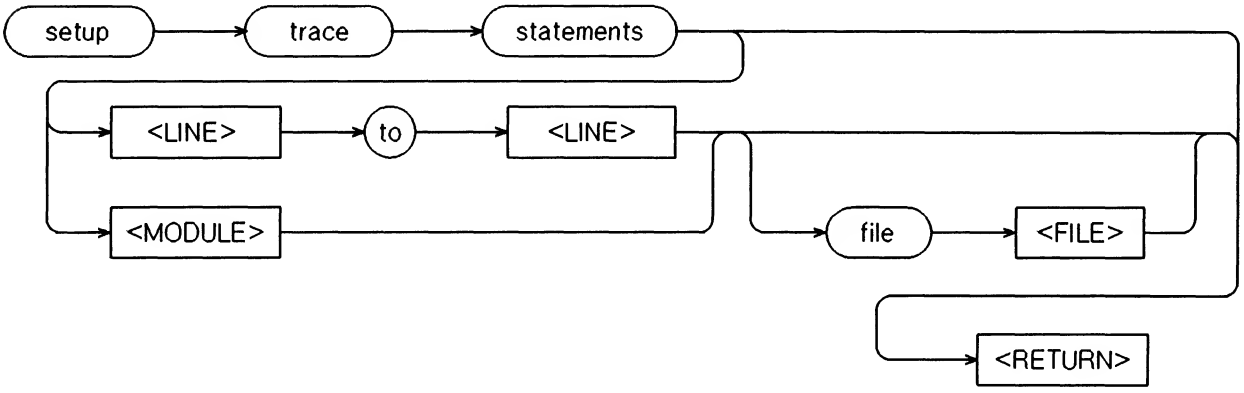


Figure 8-7. Setup Trace Statements Syntax Diagram

trace statements
(Cont'd)

Parameters

The following definitions describe the parameters used in the ***setup trace statements*** command.

file	***file*** indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the ***file*** softkey.
<FILE>	<FILE> is an optional parameter that refers to the source file containing the specified <MODULE> or line range called out in the command statement. If the <MODULE> or line range is in the defined default path, the <FILE> parameter may be omitted from the command statement.
<LINE>	<LINE> represents the line number of a Pascal or C statement in the source program. The two line numbers specified are the boundaries of the trace measurement. The first line is inclusive and is traced. The second line (following the ***to*** in the measurement specification) is noninclusive and is not traced. If the specified <LINE> contains only comments (no executable code), the analyzer will associate the line number with the first line containing executable code following it. Any comment lines preceding the first line of executable code in a procedure or function are not recognized by the software analyzer. All lines in the specified line range must be contained within a single module. This module may be a procedure or function, or the main program block.
<MODULE>	<MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure or a function within a specified file. In C, a module can be the name of a function within a specified file. The trace statements measurement traces source level statements and all variables referenced in the source statements contains in the specified <MODULE>.

Setup Command Examples

The following command examples illustrate how to use the ***setup trace statements*** command to define measurements.

setup trace statements PROC2 ***file*** TESTP
setup trace statements 74 ***to*** 102

Trace Statements Measurement Example

The following example shows several lines of a program, a setup trace statements command for the program segment, and the resulting trace display.

trace statements (Cont'd)

SOURCE PROGRAM LINES. The following source program segment is traced in the trace statements measurement example.

```
68   FUNCTION PROC1(VAR SN:INTEGER; SV:INTEGER):INTEGER;
69     VAR D:INTEGER;
70         MESSY:REC_TYPE_PTR;
71
72     PROCEDURE SETUP(VAR SNN:INTEGER; SNV:INTEGER; VAR SVN:INTEGER;
73       SVV:INTEGER; VAR DN:INTEGER; DV:INTEGER;VAR PTR:REC_TYPE_PTR);
...
80       BEGIN  (*SETUP MAIN BODY*)
81
82         PTR^.I := PTR^.I-1;
83         Y:=1;
84         D:=D-1; (*Scoped variable*)
85         P2:=SNN; (*STATIC  CALLBYNAME  CALLBYNAME*)
86         P2:=SNN; (*STATIC  CALLBYNAME  CALLBYNAME*)
87         P2:=SNV; (*STATIC  CALLBYNAME  CALLBYVALU*)
88         P2:=SVN; (*STATIC  CALLBYVALU  CALLBYNAME*)
89         P2:=SVV; (*STATIC  CALLBYVALU  CALLBYVALU*)
90         P2:=DV;  (*        DYNAMIC     CALLBYVALU*)
91         P2:=DN;  (*        DYNAMIC     CALLBYNAME*)
92         COLOR_SET:= [ BLACK,BROWN ];
93
94         IF COUNT = 10
95           THEN  COUNT:=0
96           ELSE
97             BEGIN
98               COUNT := COUNT+1;
99               COLOR_SET := COLOR_SET + [ WHITE, GREEN ];
100              SETUP(SNN,SNV,SVN,SVV,DN,DV,PTR);
101            END;
...
117    BEGIN  (*PROC1 MAIN BODY*)
...
123      SETUP  (SN,SN,SV,SV,D,D,MESSY^.NEXT_REC);
...
127    END;   (*PROC1 MAIN BODY*)
...
151  BEGIN (*MAIN PROGRAM*)
...
182    X:=PROC1(COUNT,COUNT+2);
...
205  END. (*MAIN PROGRAM*)
```

SETUP MEASUREMENT COMMAND. The following ***setup trace statements*** command results in the setup display shown in figure 8-8.

setup trace statements SETUP ***file*** MAIN

trace statements (Cont'd)

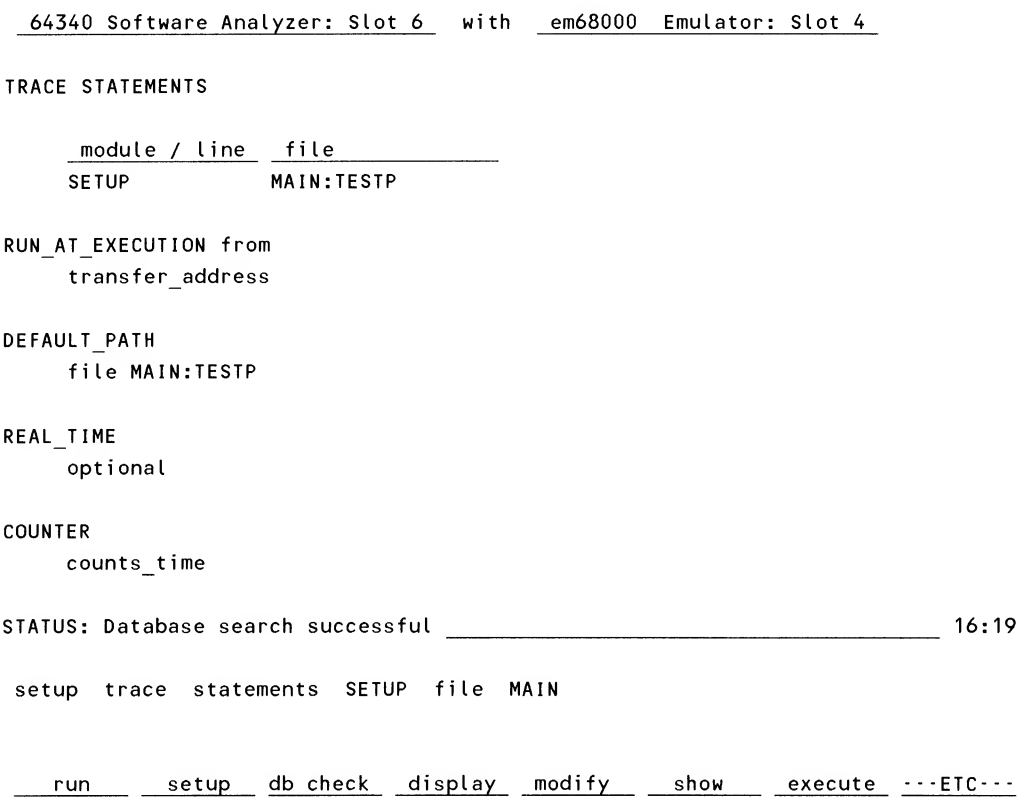


Figure 8-8. Trace Statements Setup Display

MEASUREMENT DISPLAY. Figure 8-9 is a ***trace statements*** measurement listing showing the source lines that were executed and and the values of the variables accessed or modified in the source lines. "Break for new stack information" indicates that the analyzer has started tracing a different occurrence (activation) of the procedure.

Accessed Variables Not Traced. Some variables accessed in a source line are not traced. This includes variables that are maintained in registers rather than in memory. This may occur if the compiler AMNESIA option is off. In the C programming language, array parameters without an explicitly defined size are not traced. The value of a pointer variable is traced but the object of the pointer is not traced.

Effects of Prefetch. The following limitations apply when the analyzer is used with a target processor which has an instruction prefetch mechanism and an emulator that does not dequeue the prefetch.

1. Symbols may not line up with the source line that accessed them. This is seen in the sample display in figure 8-9.

trace statements (Cont'd)

2. Executed source lines may not be displayed if the number of program fetches for the source line is less than the depth of the prefetch queue.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Source                                    Symbol      Value       Stat   Time-rel

Break for new stack information
  82 PTR^.I := PTR^.I-1;                  PTR          000003028H read      0.0 uS
  83 Y:=1;                                                                 4.3 uS
  84 D:=D-1; (*Scoped variable*)          Y          1.00000E0 write   481.4 uS
  85 P2:=SNN; (*STATIC  CALLBYNAME  *     D                  4 read      4.2 uS
                                          D                  3 write     2.4 uS
                                          SNN                2 read      3.6 uS
  86 P2:=SNN; (*STATIC  CALLBYNAME  *     P2                 2 write     1.3 uS
                                          SNN                2 read      5.0 uS
  87 P2:=SNV; (*STATIC  CALLBYNAME  *     P2                 2 write     1.2 uS
                                          SNV                2 read      3.4 uS
  88 P2:=SVN; (*STATIC  CALLBYVALU  *     P2                 2 write     1.0 uS
                                          SVN                4 read      5.1 uS
  89 P2:=SVV; (*STATIC  CALLBYVALU  *     P2                 4 write     1.0 uS

STATUS: Awaiting Command                                        36    16:12


  run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 8-9. Trace Statements Measurement Display (Real-Time Optional)

In a prefetch environment, the source line may be off by plus or minus one line or variables may be displayed with the wrong source line.

Source Line Blanking. Whenever statements are repeated sequential on the screen, the source line is displayed only for the first occurrence of the statement. For the 2nd through nth occurrence of the line, only a quote mark " " " is displayed in the source field. Statements may be repeated for the following reasons:

1. The statement causes multiple data accesses.
2. The statement contains an implicit loop (e.g., assignment of a large data structure).
3. The statement contains an explicit loop (e.g., WHILE FLAG TRUE DO...).

trace statements
(Cont'd)

Real-Time Optional Vs. Real_Time Required. The trace statements measurement can be made in either real-time optional or real-time required modes. However, executing a measurement with real_time required mode selected, only static variables can be acquired and displayed. The effects on the measurement can be seen by comparing the display in figure 8-10 with that in figure 8-9. Note that no dynamic variables were captured in real_time required mode (figure 8-10).

Unexpected Symbol Names Displayed In Real-Time Required Mode. When executing a trace statements measurement in real_time required mode, you may see unexpected symbol names displayed in the symbol field of the measurement display. This occurs if a static variable defined at the program level is passed by reference to the procedure or function being traced. Since the analyzer does not have access to the variable name declared within the procedure (breaking program execution to read the stack frame is not allowed in real-time required mode), the global symbol assigned to the parameter address passed to the procedure or function is used.

This effect can be seen in figure 8-10. the symbol COUNT is displayed in the symbol field for source program lines 85 and 86, although the symbol being read in the source line is SNN. Referring back to the source program listing, we see that SNN is a pass-by-reference parameter (line 72). The parameter name passed to procedure SETUP from Function PROC1 is SN (see the procedure call at line 123). Similarly SN is a pass-by-reference parameter to PROC1 (line 68). In the calling statement to PROC1 (line 182), we see that the main program passed the global variable COUNT to PROC1. This is the symbol displayed in the symbol field.

Note that the Symbol SNN is display in the trace listing in figure 8-9. Since this trace was executed in real_time optional mode, the analyzer executed a break at the start of the procedure to read the stack frame information. This enabled the analyzer to access variables local to the procedure.

trace statements

(Cont'd)

```
64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Source                                    Symbol      Value       Stat   Time-rel

  82 PTR^.I := PTR^.I-1;                                                   0.0 uS
  83 Y:=1;                                                                 3.8 uS
  84 D:=D-1; (*Scoped variable*)                                         482.0 uS
  85 P2:=SNN; (*STATIC  CALLBYNAME  * COUNT                 2 read       4.7 uS
  86 P2:=SNN; (*STATIC  CALLBYNAME  * COUNT                 2 read       6.7 uS
  87 P2:=SNV; (*STATIC  CALLBYNAME  *                                      6.1 uS
  88 P2:=SVN; (*STATIC  CALLBYVALU  *                                      4.4 uS
  89 P2:=SVV; (*STATIC  CALLBYVALU  *                                      6.1 uS
  90 P2:=DV;  (*        DYNAMIC     *                                      4.5 uS
  91 P2:=DN;  (*        DYNAMIC     *                                      4.5 uS
  92 (* IS A OF COLOR_SET*)(* A B C**                                      5.7 uS
  94 IF COUNT = 10                  COUNT               2 read           3.4 uS
  98 COUNT := COUNT+1;                                                     6.5 uS
  99 COLOR_SET := COLOR_SET + [ WHIT* COUNT             2 read           1.1 uS

STATUS: Awaiting Command                                          36    16:12


   run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 8-10. Trace Statements Measurement Display (Real-Time Required)

TRACE STATEMENTS DON'T CARE DISPLAY. A example trace statements "don't care" display is shown in figure 8-11. In a "don't care" measurement, the analyzer may trace states in the user program which do not correspond to source statements or states that are outside the user program. Note the two display lines immediately following line 100 in the display. The message "**No source line found (PC= 00001...)**" is displayed, indicating that these lines are overhead generated by the compiler for procedure entry. The message "**???? File not found, file = ...**" is displayed on the lines following source line 83. This message is displayed when states are traced in a file for which there is no database. The file name is displayed with the PC value executed. In this case, the file is the library routine SFLOAT:LR68K. Other messages may be displayed in a trace statements "don't care" display. See Appendix B for an explanation of status and error messages.

NOTE

Do not use ***trace statements don't care*** in real_time optional mode. This measurement should always be executed in real_time required mode.

trace statements
(Cont'd)

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Source                                                          Source Path

  94 IF COUNT = 10                                              NT1:TESTP
  98 COUNT := COUNT+1;                                          NT1:TESTP
  99 COLOR_SET := COLOR_SET + [ WHITE, GREEN ];                 NT1:TESTP
 100 PROC2(SNN,SNV,SVN,SVV,DN,DV,PTR);                         NT1:TESTP
???? No source line found (PC= 000001000H)                     NT1:TESTP
???? No source line found (PC= 000001002H)                     NT1:TESTP
  82 PTR^.I := PTR^.I-1;                                        NT1:TESTP
  83 Y:=1;                                                      NT1:TESTP
???? File not found, file= SFLOAT:LR68K:comp_db (PC= 000002* SFLOAT:LR68K
???? File not found, file= SFLOAT:LR68K:comp_db (PC= 000002* SFLOAT:LR68K
???? File not found, file= SFLOAT:LR68K:comp_db (PC= 000002* SFLOAT:LR68K
???? File not found, file= SFLOAT:LR68K:comp_db (PC= 000002* SFLOAT:LR68K
???? File not found, file= SFLOAT:LR68K:comp_db (PC= 000002* SFLOAT:LR68K
???? File not found, file= SFLOAT:LR68K:comp_db (PC= 000002* SFLOAT:LR68K

STATUS: Awaiting Command                                    52    12:04


   run     setup   db check   display   modify    show     execute   ---ETC---
```

Figure 8-11. Trace Statements Don't Care Display (Real-Time Required)

trace variables

The trace variables measurement allows you to trace specified variables and parameters and display their values, along with the source statement that accessed them. The variables are displayed in their declared data type format, i.e., as integers, reals, boolean values, characters, etc. The variable must be uniquely defined as to the module where it is declared. If the variable is defined outside a module, i.e., a program variable in Pascal or an outer level variable in C, then only the file name is required. When multiple variables map to the same memory location, only the first variable specified in the setup command is displayed.

A maximum of 10 adjacent symbols or 9 non-adjacent symbols may be traced.

Command Syntax

The command syntax for setting up the ***trace variables*** measurement is shown in figure 8-12.

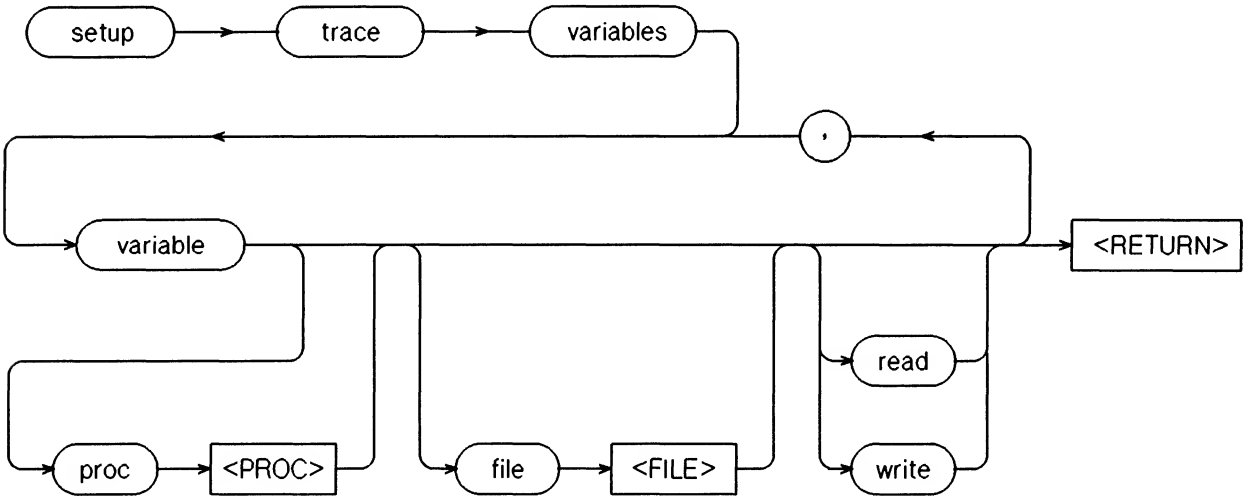


Figure 8-12. Setup Trace Variables Syntax Diagram

Parameters

The following paragraphs define the parameters used in the ***setup trace variables*** command.

file	***file*** indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the ***file*** softkey.
<FILE>	<FILE> is an optional parameter that refers to the source file containing the specified <VAR> or <PROC> called out in the command statement. If the <VAR> or <PROC> is in the defined default path, the <FILE> parameter may be omitted from the command statement.

trace variables (Cont'd)

<INDEX> Represents an index value (integer or scalar value) specifying a component of an array.

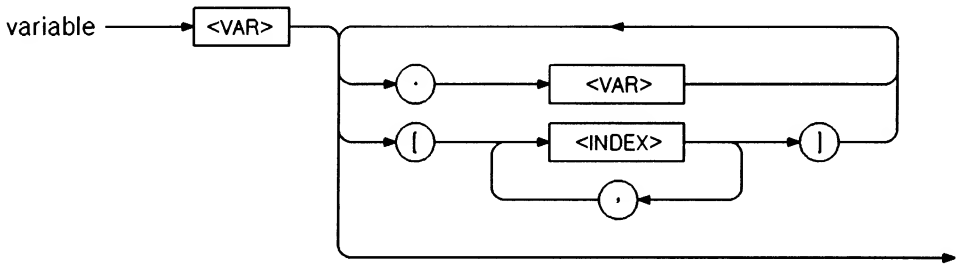
proc ***proc*** indicates that a procedure or function name follows that defines the procedure or function to which a variable belongs. NOTE: an "@" may be used in place of ***proc***.

<PROC> <PROC> is an optional parameter that refers to a procedure or function in which <VAR> is declared. If <PROC> is defined in the ***setup default_path*** command, it may be omitted in the ***setup trace variables*** command. If <PROC> is not specified in either the default path or the ***setup trace variables*** command, the analyzer assumes that <VAR> is a variable defined at the main program level.

read ***read*** specifies that only memory read accesses to the specified variable be traced. The default condition is to trace both memory read and memory write accesses to the specified variable.

<VAR> <VAR> represents the name of a variable or parameter to be traced. <VAR> can be any valid Pascal or C variable expression. Pointer variables cannot be traced in the trace variables measurement mode.

variable Variable may be any valid C or Pascal variable other than pointer types. Pointer variables cannot be traced with the ***trace variables*** measurement.



write ***write*** specifies that only memory write accesses to the variable be traced.

trace variables (Cont'd)

Setup Command Examples

The following command examples illustrate how to use the ***setup trace variables*** command to define measurements.

setup trace variables pred_result ***proc*** proc2
setup trace variables COUNT , SNN ***proc*** PROC2 , Q.FLAG ***file*** TESTP

Trace Variables Measurement Example

The following example shows several lines of program, a setup trace variables command and the resulting trace display.

SOURCE PROGRAM LINES. The following source program segment is traced in the trace variables measurement example.

```
166    pred_result.enumerated = green
167    check = check + pred_result.arr[0]
168    check = check - result.arr[0];
169    check = check + pred-result.arr[1];
170    check = check - result.arr[1];
171    pred_result.enumerated = blue
172
173    u16();
174    pred_result.ch = 'a';
175    check = check + pred_result.arr[0];
176    check = check - result.arr[0];
177    check = check + pred_result.arr[1];
178    check = check - result.arr[1];
179    pred_result.ch = 'a';
```

trace variables
(Cont'd)

SETUP MEASUREMENT COMMAND. The following ***setup trace variables*** command specifies that the software analyzer trace all occurrences of variable pred_results.

setup trace variables pred_result

Figure 8-13 show setup display resulting for executing the setup command.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

TRACE VARIABLES

     variable                proc          file            status
     pred_result                           P_CHECK:TESTP   read_write

RUN_AT_EXECUTION from
     transfer_address

DEFAULT_PATH
     file P_CHECK:TESTP

REAL_TIME
     optional

COUNTER
     counts_time

STATUS: Database search successful                                    16:19

 setup  trace  variables  pred_result


  run    setup  db check  display  modify   show    execute  ---ETC---
```

Figure 8-13. Trace Variables Setup Display

trace variables (Cont'd)

MEASUREMENT DISPLAY. The trace list in figure 8-14 shows all accesses to the variable pred_result where pred_result is a structure. The value of the variable and the source line from which it was accessed are shown.

The read and subsequent write of pred_result.u8 at source line 28 is due to the read then write nature of the instruction used by the target processor to clear a memory location.

pred_result.s16 is set to hexadecimal value -1BFE on line 39 in the source field but is displayed as decimal value -7166 in the value field. The default base for numeric data types is decimal.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

symbol                 Value        stat  source
pred_result.enumera*         red  write   17 pred_result.enumerated = red;
pred_result.arr[0]             0  read   158 check = check + pred_result.arr[0]
pred_result.u8                50  read    28 pred_result.u8 = 0;
pred_result.u8                 0  write   28 pred_result.u8 = 0;
pred_result.arr[0]             0  read   162 check = check + pred_result.arr[0]
pred_result.s16            -7166  write   39 pred_result.s16 = -1BFEH;
pred_result.enumera*       green  write  166 pred_result.enumerated = green;
pred_result.arr[0]             0  read   167 check = check + pred_result.arr[0]
pred_result.arr[1]             0  read   169 check = check + pred_result.arr[1]
pred_result.enumera*        blue  write  171 pred_result.enumerated = blue;
pred_result.ch               "A"  write  174 pred_result.ch = 'A';
pred_result.arr[0]             0  read   175 check = check + pred_result.arr[0]
pred_result.arr[1]             0  read   177 check = check + pred_result.arr[1]
pred_result.ch               "a"  write  179 pred_result.ch = 'a';
pred_result.s16             3700  write   85 pred_result.s16 = 3700;

STATUS: Awaiting Command                                          30    16:12


   run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 8-14. Trace Variables Measurement Display

NOTES

Chapter 9

MAKING COUNT AND TIME MEASUREMENTS

OVERVIEW

This chapter describes the following software analyzer measurements.

- count statements
- time modules

GENERAL INFORMATION

The Count Statements and Time Modules measurements allow you to perform coverage testing and performance analysis of software modules. The Count Statements measurement shows the number of times a source statement or range of source statements are executed. The Time Modules measurement measures up to four modules for real-time execution speed, identifying bottlenecks that may require recoding of modules.

count_statements

The Count Statements measurement enables you to measure the number of times that selected source code lines are executed. You may specify a range of up to 255 source code lines to be counted within a single software module. The measurement display shows the specified source lines along with the number of times each source line was executed.

Syntax

The command syntax for setting up a Count Statements measurement is shown in figure 9-1.

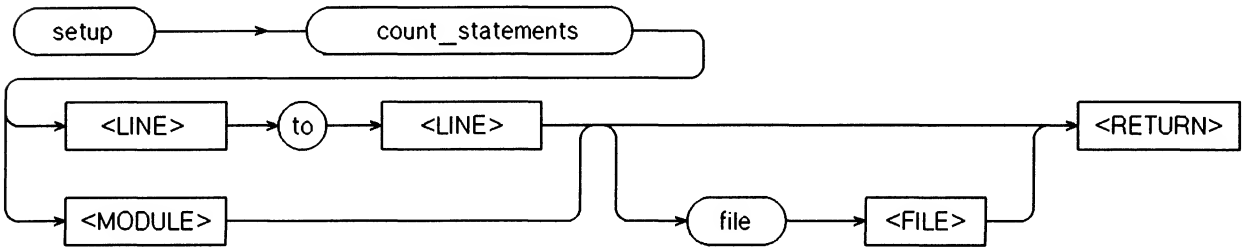


Figure 9-1. Setup Count_Statements Command Syntax

Parameters

file — *file* indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the *file* softkey.

<FILE> — <FILE> specified the source file to be used as the default path. When the <FILE> parameter is omitted from a measurement command, the file specified as the default path is used.

<LINE> — <LINE> represents the line number of a Pascal or C statement in the source program. Any comment lines preceding the first line of executable code in a procedure or function are not recognized by the software analyzer. All lines in the specified line range must be contained within a single module. This module may be a procedure or function in Pascal or a function in C, or the main program block.

<MODULE> — <MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure, function or the main program within a specified file. In C, a module can be the name of a function within a specified file.

to — *to* is used to specify a line range to be counted in a source program. All lines in the specified range must be contained in a single module. The total number of lines must not exceed 255, and the total address space the range covers cannot exceed 4096.

count_statements (Cont'd)

Setup Command Examples

setup count_statements 112 *to* 131 *file* SORT
setup count_statements MATRIX

Count Statements Measurement Example

The following example shows several lines of a program, a *setup count_statements* command for the program segment, and the resulting setup and measurement displays.

SOURCE PROGRAM LINES. The following source program segment is measured in the count_statements measurement example.

```
94    IF COUNT = 10
95      THEN  COUNT:=0
96      ELSE
97        BEGIN
98          COUNT := COUNT+1;
99          COLOR_SET := COLOR_SET + [ WHITE, GREEN ];
100         PROC2(SNN,SNV,SVN,SVV,DN,DV,PTR);
101       END;
```

SETUP MEASUREMENT COMMAND. The following *setup count statements* command results in the setup display shown in figure 9-2. The measurement will count the number of times each statement in the IF.. THEN.. ELSE compound statement is executed.

setup count_statements 94 *to* 100

count statements
(Cont'd)

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

COUNT STATEMENTS

     module / line   file
     94  to 100      NT1:TESTP

ENABLE off

DISABLE off

RUN_AT_EXECUTION from
     transfer_address

DEFAULT_PATH
     file NT1:TESTP

REAL_TIME

STATUS: Database search successful                                    16:19

 setup  count_statements  94  to  100


  run    setup  db check  display  modify   show    execute  ---ETC---
```

Figure 9-2. Count Statements Setup Display

MEASUREMENT DISPLAY. Figure 9-3 shows the measurement display resulting from the preceding setup specification. From looking at the source program listing, we expect that the ELSE statement will be executed 10 times for each execution of the THEN statement. In addition, the number of times the IF statement is executed should be equal to the sum of the executions of the ELSE and THEN statements. The count for each statement in the display verifies that the IF.. THEN.. ELSE statement did execute properly.

count_statements (Cont'd)

```
64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Count-abs Source

      682   94 IF COUNT = 10
       62   95 THEN  COUNT:=0
        0   96 ELSE
        0   97 BEGIN
      620   98 COUNT := COUNT+1;
      620   99 COLOR_SET := COLOR_SET + [ WHITE, GREEN ];


STATUS: Awaiting command                                    0 __ 13:09

  run    setup   db check   display   modify    show    execute  ---ETC---
```

Figure 9-3. Count Statements Measurement Display

time_modules

The Time Modules measurement shows the execution time of software modules executing at the processor's full operating speed. The Time Modules measurement times up to four modules simultaneously. Both single and multiple measurements may be made. If a module occurs more than once during a measurement, measurement statistics (minimum, maximum, and mean execution time, and number of occurrences of the module) are displayed automatically. The measurement can trace up to 256 levels of recursion.

Syntax

The command syntax for setting up a time modules measurement is shown in figure 9-4.

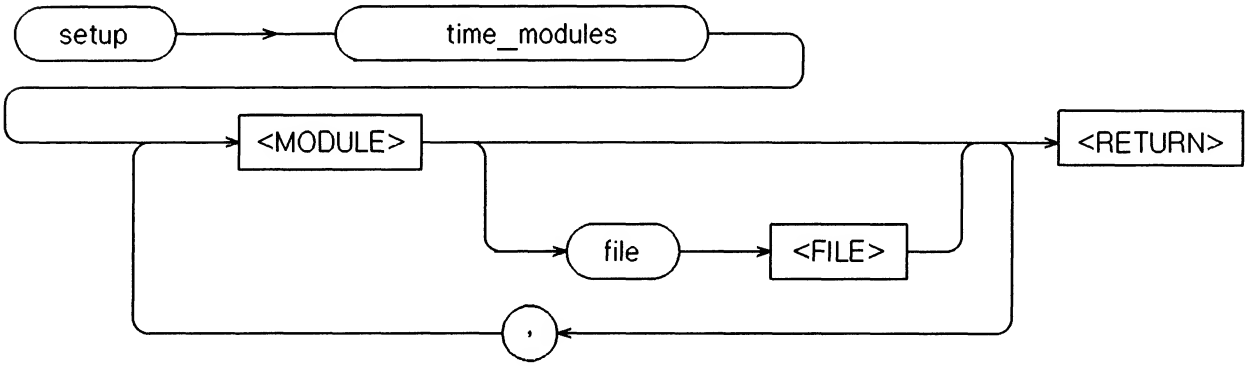


Figure 9-4. Setup Time_Modules Command Syntax

Parameters

file	*file* indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the *file* softkey.
<FILE>	<FILE> specified the source file to be used as the default path. When the <FILE> parameter is omitted from a measurement command, the file specified as the default path is used.
<MODULE>	<MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure function or the main program within a specified file. In C, a module can be the name of a function within a specified file.

Examples

setup time_modules SORT_ELEMENTS *file* SORT
setup time_modules MATRIX , BUB_SORT

time_modules (Cont'd)

Time modules Measurement Example

The following example shows a setup *time_modules* command with the resulting setup and measurement displays.

SETUP MEASUREMENT COMMAND. The following *setup time_modules* command results in the setup display shown in figure 9-5. The measurement will measure the execution times of modules PROC1, PROC2, PROC4, and PROC10.

setup time_modules PROC1 , PROC2 , PROC4 , PROC10

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

TIME MODULES

     module          file
     PROC1           NT1:JGREEN
     PROC2           NT1:JGREEN
     PROC4           NT1:JGREEN
     PROC10          NT1:JGREEN

ENABLE off

DISABLE off

RUN_AT_EXECUTION from
     transfer_address

DEFAULT_PATH

STATUS: Database search successful ____________________________________ 16:19

 setup  time_modules  PROC1 , PROC2 , PROC4 , PROC10


  run    setup  db check  display  modify   show    execute  ---ETC---
```

Figure 9-5. Time Modules Setup Display

MEASUREMENT DISPLAY. Figure 9-6 shows the measurement display resulting from the preceding setup specification. The display shows the minimum, maximum and mean execution time for the specified modules and the number of times (count field) each module was executed.

time_modules
(Cont'd)

```
64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

  Minimum    Maximum       Mean  Count Symbol          Symbol path

 47.21 mS   59.90 mS   59.79 mS    152 PROC1           PROC1:NT1:TESTP
 6.289 mS   59.04 mS   36.32 mS   1670 PROC2           PROC2:NT1:TESTP
 59.89 mS   59.92 mS   59.90 mS     75 PROC4           PROC4:NT1:TESTP
   8.8 uS     9.2 uS     9.0 uS     75 PROC10          PROC10:NT1:TESTP

STATUS:  Execution complete  (saved =    8)                          13:09

   run     setup   db check   display   modify    show    execute  ---ETC---
```

Figure 9-6. Time Modules Measurement Display

Chapter 10

USING INTERACTIVE COMMANDS FOR PROGRAM DEBUGGING

OVERVIEW

This chapter describes the following software analyzer measurements that allow you to interact with the HP 64000 emulation system.

- Setup break
- Display <VAR>
- Modify <VAR>

GENERAL INFORMATION

The software analyzer has three commands that allow you to interact with the emulator without exiting the analyzer. These commands are ***setup break***, ***display <VAR>***, and ***modify <VAR>***. Detailed descriptions of how to use these commands are given in this chapter.

setup break

The *setup break* command provides you with two separate functions. The *break on measurement_complete* command is a setup specification that specifies that the analyzer is to break program execution on completion of the specified measurement. The *break on <LINE>* or *break on <MODULE>* is a measurement that breaks program execution at the specified program location.

By defining and executing a series of breakpoints you can locate a position in the program under test to a combination of sequential events. Up to nine hardware breakpoints may be defined in the *setup break* command.

Command Syntax

The command syntax for defining hardware breaks is shown in figure 10-1.

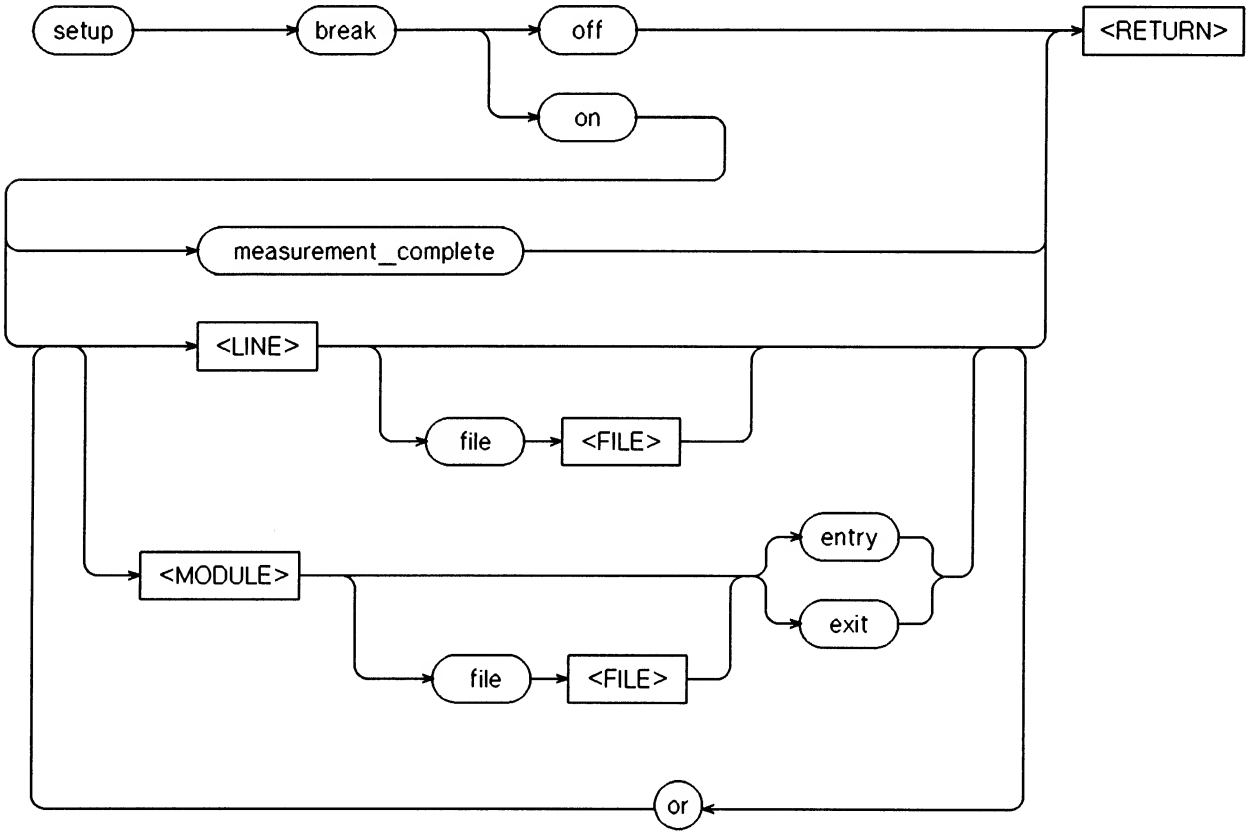


Figure 10-1. Setup Break Syntax Diagram

setup break (Cont'd)

Default Value

setup break off (affects ***break on measurement_complete*** only)

Parameters

The following definitions describe the parameters used in the ***setup break*** command.

entry	***entry*** defines the breakpoint to be the entry point to the specified module.
exit	***exit*** defines the breakpoint to be the exit point from the specified module.
file	***file*** indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the ***file*** softkey.
<FILE>	<FILE> is an optional parameter that refers to the source file containing the specified <MODULE> or line called out in the command statement. If the file containing the <MODULE> or line is the defined default path, the <FILE> parameter may be omitted from the command statement.
<LINE>	<LINE> represents the line number of a Pascal or C statement in the source program. If the specified <LINE> contains only comments (no executable code), the analyzer will associate the line number with the first line containing executable code following it. Any comment lines preceding the first line of executable code in a procedure or function are not recognized by the software analyzer.
<MODULE>	<MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure or a function within a specified file. In C, a module can be the name of a function within a specified file.
off	***off*** disables the ***setup break on measurement_complete*** command.
on	***on*** allows you to define the conditions on which to break.

Setup Command Examples

The following command examples illustrate how to use the ***setup break*** command to define measurements.

setup break on PROC2 ***entry***
setup break on 102
setup break on measurement_complete
setup break on RECURSIVE_PROC ***exit or*** 134

display <VAR>

The ***display <VAR>*** command displays the current value of a variable in memory. The variable is displayed in the data type that they were declared in the Pascal or C source file, i.e., as integers, reals, boolean values, characters, etc. The variable must be uniquely defined as to the module where it is declared. If the variable is defined outside of a module, i.e., a program variable in Pascal or outer level variable in C, then only the file name is required. The user program must be halted and the emulator running in the emulation monitor before the ***display<VAR>*** command can be executed. The variable to be displayed must be accessible based upon the next address the program will execute. Local variables are accessible only if (1) the next program counter is within the user code of the procedure that defined the variable or (2), the variable belongs to a parent procedure of the current executing procedure.

Command Syntax

The command syntax for the ***display <VAR>*** command is shown in figure 10-2.

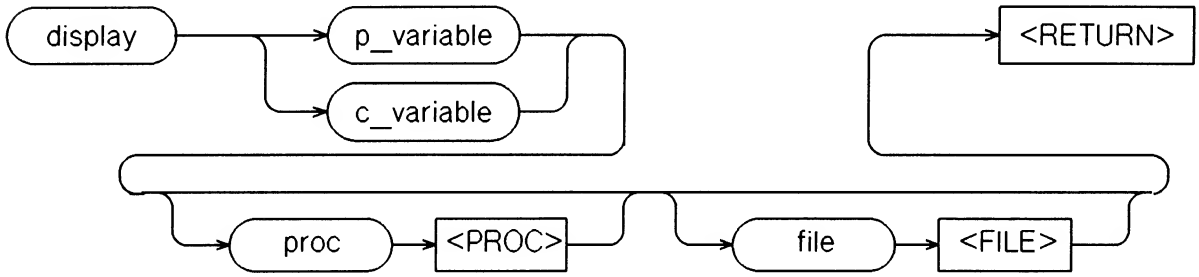
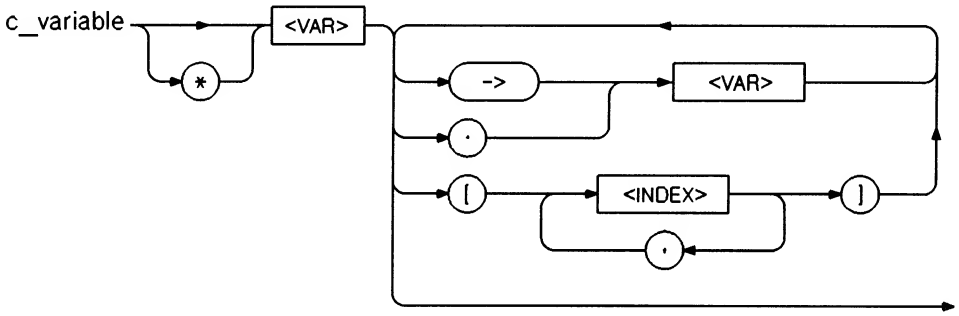


Figure 10-2. Display Variables Syntax Diagram

Parameters

The following paragraphs define the parameters used in the ***display <VAR>*** command.

c_variable — c_variable may be any valid C variable in the following expression format.



file — ***file*** indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the ***file*** softkey.

display <VAR> (Cont'd)

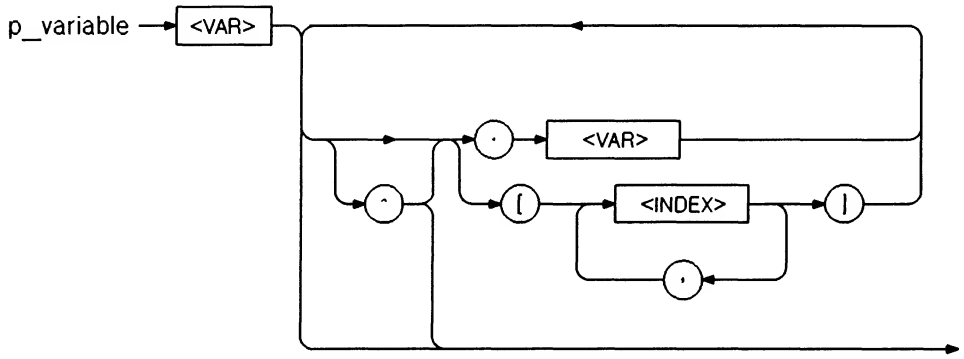
<FILE> — <FILE> is an optional parameter that refers to the source file containing the specified <VAR> and <PROC> called out in the command statement. If the <VAR> and <PROC> is in the defined default path, the <FILE> parameter may be omitted from the command statement.

<INDEX> — Represents an index value (integer or scalar value) specifying a component of an array.

proc — ***proc*** indicates that a procedure or function name follows that defines the procedure or function to which a variable belongs. NOTE: an "@" may be used in place of ***proc***.

<PROC> — <PROC> is an optional parameter that refers to a procedure or function in which <VAR> is declared. If <PROC> is defined in the ***setup default_path*** command, it may be omitted in the ***display*** command. If <PROC> is not specified in either the default path or the ***display*** command, the analyzer assumes that <VAR> is a variable defined at the main program level.

p_variable — p_variable may be any valid Pascal variable in the following expression format.



<VAR> — <VAR> represents the name of a variable or parameter to be displayed. <VAR> can be any valid Pascal or C variable expression.

Display Command Examples

The following command examples illustrate how to use the ***display <VAR>*** command to display the value of variables.

display SNN^ ***proc*** PROC2 ***file*** NT1
display Q.FLAG ***proc*** CONTROLT
display A[1] ***file*** TESTP
display A^.B^.C^
display *a->b->c

modify <VAR>

The ***modify <VAR>*** command allows you to modify the current value of variables in memory. Values must be specified in binary, octal, decimal, or hexadecimal notation. The variable must be uniquely defined as to the module where it is declared. If the variable is defined outside of a module, i.e., a program variable in Pascal or an outer variable in C, then only the file name is required. The user program must be halted and the emulator running in the emulation monitor before the ***modify <VAR>*** command can be executed. The maximum variable size that can be modified with a single command is 32 bits. Larger variables must have their subelements modified individually with multiple commands. The variable must be accessible based upon the next address the program will execute. Local variables are accessible only if (1) the next program counter is within the user code of the procedure that defined the variable or (2), the variable belongs to a parent procedure of the current executing procedure.

Command Syntax

The command syntax for the ***modify <VAR>*** command is shown in figure 10-3.

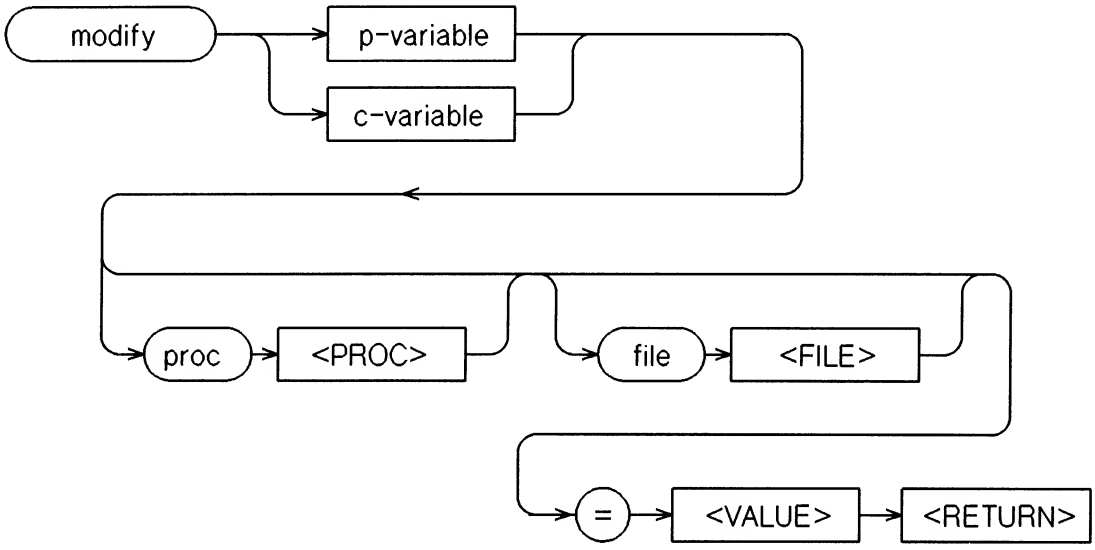


Figure 10-3. Modify <VAR> Syntax Diagram

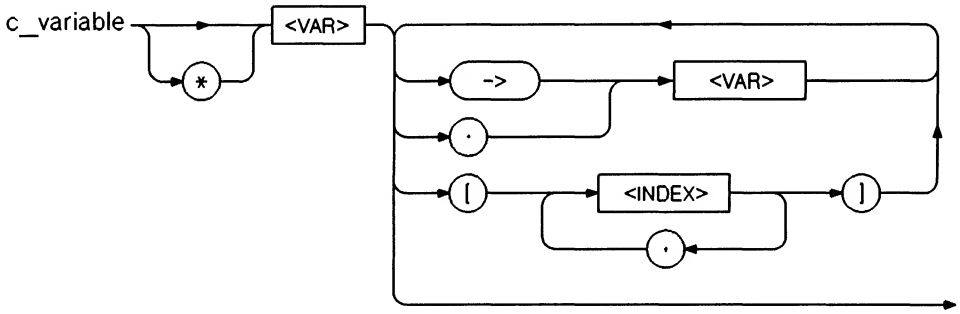
modify <VAR> (Cont'd)

Parameters

The following paragraphs define the parameters used in the ***modify*** command.

c_variable

c_variable may be any valid C variable in the following expression format.



file

file indicates that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the ***file*** softkey.

<FILE>

<FILE> is an optional parameter that refers to the source file containing the specified <VAR> and <PROC> called out in the command statement. If the <VAR> and <PROC> is in the defined default path, the <FILE> parameter may be omitted from the command statement.

<INDEX>

Represents an index value (integer or scalar value) specifying a component of an array.

proc

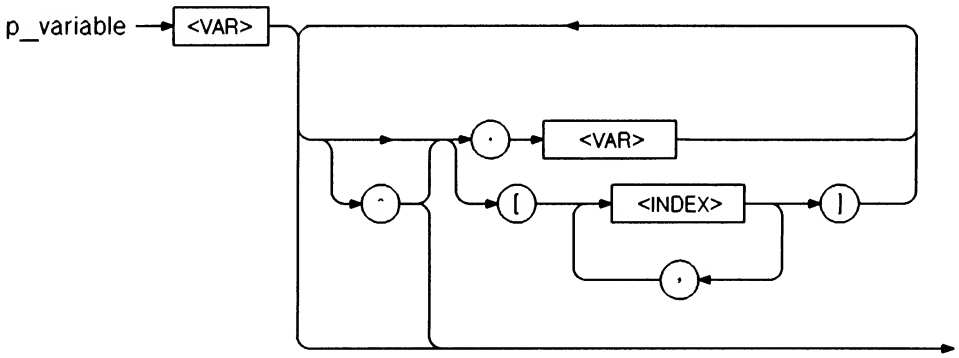
proc indicates that a procedure or function name follows that defines the procedure or function to which a variable belongs. NOTE: an "@" may be used in place of ***proc***.

<PROC>

<PROC> is an optional parameter that refers to a procedure or function in which <VAR> is declared. If <PROC> is defined in the ***setup default_path*** command, it may be omitted in the ***display*** command. If <PROC> is not specified in either the default path or the ***modify*** command, the analyzer assumes that <VAR> is a variable defined at the main program level.

modify <VAR>
(Cont'd)

p_variable — p_variable may be any valid Pascal variable in the following expression format.



<VALUE> — <VALUE> represents the value that the specified variable is to be changed to. <VALUE> must be specified as an integer value.

<VAR> — <VAR> represents the name of a variable or parameter to be modified. <VAR> can be any valid Pascal or C variable expression.

Modify Command Examples

The following command examples illustrate how to use the ***modify*** command to change the value of program variables.

```
modify Q.CHAR1 proc LTRSORT = 41H
modify NEXTINT = 0124H
modify A^.B^.C^ = 15
modify *a->b->c = 15
```

Chapter 11

MAKING INTERMODULE BUS MEASUREMENTS

OVERVIEW

This chapter describes the intermodule measurement capabilities of the Real-Time High Level Software Analyzer. The following topics are covered in this chapter:

- Intermodule bus signals
- Interaction between the software analyzer and the IMB
- Software analyzer trigger enable command
- Driving trigger enable with the software analyzer
- Receiving trigger enable from another analysis module

INTRODUCTION

Intermodule measurements are measurements involving two or more analysis modules. Intermodule measurements are coordinated between analysis modules by means of a high speed intermodule bus (IMB). The IMB coordinates triggering, windowing of functions, and synchronization of execute and halt commands for all modules involved in a measurement via the intermodule bus cable connected to the IMB connectors on each analyzer control board.

When the HP 64000 development station power is switched on, there is no intermodule specification between the software analyzer and other analysis modules. In order to execute intermodule measurements, the software analyzer must be setup to receive or drive the intermodule trigger enable signal. The software analyzer can receive a trigger enable from other analyzers in the system. The software analyzer can also drive the IMB trigger enable line to provide a trigger enable for other analzyers.

INTERMODULE BUS SIGNALS

The software analyzer can interact with two of the intermodule bus signals, **master enable** and **trigger enable**. These signals are described in the following paragraphs. Refer to the Measurement System Reference Manual for more detailed information concerning measurement system interaction.

Master Enable

The master enable line is shared by all analysis modules included in a measurement. When master enable is true, it enables all modules that receive it. When master enable is false, it disables all modules that receive it. The master enable signal synchronizes measurement start in all analysis modules used in an IMB measurement. At the start of a measurement, all analysis modules hold the master enable line false. As each analyzer becomes ready to start, it releases the master enable line. The master enable line will go true only when the last analysis module releases the line. This starts all analysis modules synchronously.

MASTER ENABLE DRIVEN. Master enable can be driven in either one of two modes. The default mode is run synchronization, i.e., controlled by the ***execute*** and ***halt*** softkeys. In this mode, master enable remains false until all modules are ready to begin execution. Master enable then goes true and remains true until all modules have completed their measurements or until a halt command is executed. The execute and halt commands can be entered from either the measurement system level of softkeys or from within one of the analysis modules participating in the measurement.

In the other mode, master enable is driven by one designated module. Master enable still remains false until all modules are ready to begin execution, but it is controlled by the driving module once the measurement is in process. In this mode, master enable may change logical states any number of times during the measurement. The software analyzer cannot be designated as the driving module.

MASTER ENABLE RECEIVED. All modules involved in the intermodule measurement automatically receive master enable with the exception of the driver, if one is specified. The software analyzer always receives master enable when used in an IMB measurement.

Trigger Enable

The trigger enable signal windows (enables and disables) the trigger function within each module that receives it. When the trigger enable signal is true, it enables the receiving modules to recognize their triggers, if they occur. When the trigger enable signal is false, it disables trigger recognition in the receiving module. The trigger enable line can alternate between true and false during a measurement to allow the controlling analysis module to window the measurement activity in other modules where trigger recognition can occur. The software analyzer can drive the trigger enable signal or receive it from another module on the IMB.

TRIGGER ENABLE DRIVEN. Only one module can drive the trigger enable line during a measurement. If no module is designated to drive the trigger enable signal, it defaults to the true state. If more than one module is specified, the measurement cannot be executed.

TRIGGER ENABLE RECEIVED. Trigger Enable can be received by any module other than the driver.

INTERACTION BETWEEN THE SOFTWARE ANALYZER AND THE IMB

The interaction between the software analyzer and other HP 64000 system modules via the IMB (intermodule bus) is defined with the *setup trigger_enable* command, and the *setup measurement_enable* and *setup measurement_disable* commands. A measurement enable or disable condition must be defined in order to make interactive measurements over the IMB. If the enable or disable term is set to *any_state*, the IMB specification (*setup trigger_enable*) controls the measurement enable or disable function of the software analyzer. If an enable or disable term is defined, that term is combined with the *setup trigger_enable* condition to define a sequential enable or disable condition. The *any_state* parameter should be used only when making interactive measurements over the IMB. When *any_state* is specified, one state must occur before the measurement is enabled. When operating your software analyzer stand-alone (no IMB measurement specified), this may cause data to be lost at the beginning of your measurement. See Chapter 6 for detailed information about the *setup measurement_enable* and *setup measurement_disable* commands.

TRIGGER ENABLE RECEIVED. If *trigger_enable received* is specified, a trigger enable must be received from another HP 64000 analysis subsystem before the software analyzer starts looking for the measurement enable or disable condition. The trigger enable becomes the first term in a sequential measurement enable or disable condition.

TRIGGER ENABLE DRIVEN. If *trigger_enable driven_only* is specified, the software analyzer first looks for its measurement enable or disable condition. Upon finding the measurement enable or disable condition, the software analyzer drives the trigger enable line high, enabling another HP 64000 analysis subsystem, if one is set up to receive trigger enable.

SOFTWARE ANALYZER TRIGGER ENABLE COMMAND

The *setup trigger_enable* command is used to define the IMB interaction between the software analyzer and other measurement subsystems installed in the HP 64000 development station. The software analyzer must be in *real_time required* mode in order to interact with the Intermodule Bus (IMB).

Syntax

The command syntax for specifying the *trigger_enable* condition is shown in figure 11-1.

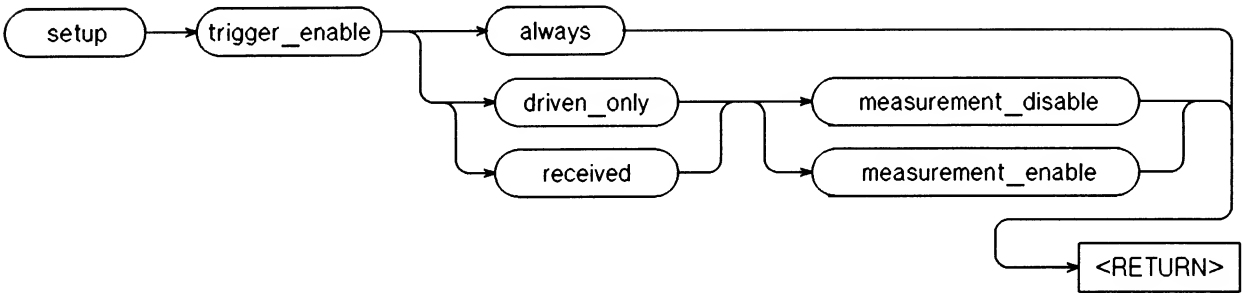


Figure 11-1. Setup Trigger_Enable Command Syntax

Default Value

always

Parameters

always — *always* specifies that *trigger_enable* is always true. This, in effect, removes the analyzer from the IMB (Intermodule Bus).

driven_only — *driven_only* specifies that the IMB trigger_enable line is to be driven on measurement_enable or measurement_disable.

measurement_disable — *measurement_disable* specifies that the IMB trigger enable line is to be driven when the specified measurement disable condition is true or received to initiate looking for the measurement disable condition. If no measurement disable condition is specified, the trigger enable command is not permitted.

measurement_enable — *measurement_enable* specifies that the IMB trigger enable line is to be driven when the specified measurement enable condition is true or received to initiate looking for the measurement enable condition. If no measurement enable condition is specified, the trigger enable command is not permitted.

received — *received* specifies that the analyzer measurement will start looking for the measurement disable or enable condition when the IMB trigger enable line is set true by another HP 64000 measurement subsystem.

Command Examples

```
setup trigger_enable always
setup trigger_enable driven_only measurement_disable
setup trigger_enable received measurement_enable
```

DRIVING TRIGGER ENABLE WITH THE SOFTWARE ANALYZER - EXAMPLE

The following measurement example illustrates how to execute an IMB measurement where the software analyzer is setup to trigger another HP 64000 analysis module. This example shows how an assembly language module called in a Pascal program can be traced. In this example, the software analyzer traces a Pascal procedure (figure 11-2) that calls an assembly language module (figure 11-3) to initialize a serial I/O port. The software analyzer is used to trigger the HP 64302A Internal Analyzer at the point in the Pascal program (line 145) where the assembly language module is called. The internal analyzer then traces execution of the assembly language module.

```
140   BEGIN                {* main routine *}
141
142     POSITION := 'T';   {initialize to position characters}
143     TOP_ROW  := ' ';   {write to leftmost of top row of LED's}
144
145     INIT_ACIA;         {set up the RS-232-C serial port}
146     HOWLONG := 220;    {initialize length of WAIT loop}
147     TEN := 10;
148
149     FOR I := 1 TO WINDOW_SIZE DO  {blank out message window in memory}
150       WINDOW[I]:=' ';
151
152     BLANK_ARRAY;       {initialize test message from DATA_AREA_0}
153     SET_ARRAY;         {set up the message for scrolling}
154     MARQUEE;           {write the message}
155
156   END.
```

Figure 11-2. Pascal Procedure PASCAL_MAIN

```
*************************************************************************
*              INITIALIZE  6850  ACIA  SERIAL  PORT                     *
*************************************************************************
               ORG             10041H
ACIA_STATUS                              ; NOTE: Both symbols at same addr
ACIA_CONTROL   DS.B            1
               ORG             10043H
ACIA_DATA      DS.B            1

               PROG

Zstartprogram  MOVE.L          #0,A5          ; Initialize heap, user
               MOVE.L          #0,A6          ;   stack pointer and
               LEA             Zstack,A7      ;   supervisory stack pointer
               JMP             [A0]           ; Jump to start of PASCAL_MAIN

INIT_ACIA
               MOVE.B          #043H,ACIA_CONTROL
*                                 ~ RESET ACIA
               MOVE.B          #00010101B,ACIA_CONTROL
*                                    ~~ DIVIDE BY SIXTEEN CLOCK
*                                  ~~~ 8 BIT DATA, 1 STOP BIT
*                                 ~ RECEIVE INTERRUPT DISABLE
RINIT_ACIA     RTS
EINIT_ACIA
               END
```

Figure 11-3. Assemble Language Module INIT_ACIA

Setting Up the Software Analyzer

To make an IMB measurement, the software analyzer must be operating in real-time mode. For this measurement, the software analyzer measurement is started by the IMB master enable signal. Therefore ***run at_execution*** is turned off. Since we wish to see the Pascal statements leading up to the assembly language module, the software analyzer is set up to trace statements in procedure PASCAL_MAIN. The measurement disable term is defined to be line 145, where we wish to trigger the HP 64302 Internal Analyzer to trace execution of the assembly language routine. The following sequence of commands set up the software analyzer for the IMB measurement:

setup real_time required

run at_execution off

setup trace statements PASCAL_MAIN

setup measurement_disable on 145

setup trigger_enable driven_only measurement_disable

Setting Up the Emulator

For all IMB measurements, emulator run control must be set up in the emulation module. If the emulator command file does not set up the emulator to receive the trigger enable signal, this must also be done in the emulator. To set up the emulator you must end out of the software analyzer and enter the emulator:

end

em68000_5

To set up the emulator to receive the trigger enable signal, enter the command:

modify configuration

Cycle through the questions until the question **Modify interactive measurement specification? no** appears on the display. Enter *yes* and press RETURN. The interactive measurement specification setup will be displayed on the HP 64000 screen.

Again cycle through the questions until **Trigger enable?** is displayed. Enter *receive* and press RETURN. Cycle through the remaining questions until you return to the emulation softkey level.

To set up emulator run control, enter the following commands:

specify run from PASCAL_MAIN

specify trace

The HP 64000 development station is now set up for the IMB measurement. At this point, you can exit the emulator and view the current measurement system configuration. See figure 11-4. Note that the software analyzer is set up to drive the trigger enable signal and the emulator is set up to receive the trigger enable signal.

```
Measurement System:  Current Configuration

 Module    Slot    Status        Description
em68000      5    completed !  Emulator for M68000
sw_anl       6    completed !  High Level Software Analyzer


Intermodule Bus      Driver(s)          Receiver(s)          | Emulation
master enable        exec/halt          em68000_5  sw_anl_6  |   Start
                                                             |
trigger enable       sw_anl_6           em68000_5            |
                                                             |
trigger                                                      |
                                                             |
storage enable                                               | Bnc Ports
                                                             | 1.
delay clock                                                  | 2.
                                                             | 3.
                                                             | 4.

STATUS: Awaiting measurement system request      userid HAIRBA _____ 13:41



em68000 5 sw anl 6  sw anly  _____  _____  print dsp  execute  end
```

Figure 11-4. Measurement System Configuration

Executing the IMB Measurement

The execute command may be given from within the emulation module, the software analyzer module, or from the the measurement system softkey level. Once the measurement is completed (or halted), you can go back and forth between the software analyzer module and the emulator module without specifying an emulation command file to view the measurement results.

The results of this measurement are shown in figures 11-5 and 11-6. Figure 11-5 shows the software analyzer trace list. Note that line 145, the call to the assemble language module and the measurement disable term, is the last line displayed. Execution of this line triggered the HP 64302A internal analyzer. The first line in the internal analyzer trace list is the beginning of the assembly language module **INIT_ACIA**.

```
64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Source                                  Symbol      Value       Stat   Time-rel

 137 END;                                                                0.0 uS
 142 POSITION := 'T';  {initialize t*                                    8.6 uS
 143 TOP_ROW  := ' ';  {write to lef* DATA_AREA_1*          54H write    2.7 uS
 145 INIT ACIA;        {set up the R* TOP ROW               " " write    2.2 uS

STATUS: Awaiting Command                                          28  --- 12:04

  run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 11-5. Software Analyzer Trace Statements Display

```
Trace:    mnemonic                        break: none          count:
 line#   address opc/data mnemonic opcode or status                time, relative
 after   001014    13FC   MOVE.B  #043H,0010041H
+001     0051F8    0000             supvr data write wd                1.    uS
+002     0051FA    1244             supvr data write wd                1.    uS
+003     001016    0043             supvr pgm  read wd                 1.    uS
+004     001018    0001             supvr pgm  read wd                 1.    uS
+005     00101A    0041             supvr pgm  read wd                <1.    uS
+006     00101C    13FC   MOVE.B  #015H,0010041H                      1.    uS
+007     010041      43             supvr data write lb                1.    uS
+008     00101E    0015             supvr pgm  read wd                 1.    uS
+009     001020    0001             supvr pgm  read wd                 1.    uS
+010     001022    0041             supvr pgm  read wd                 1.    uS
+011     001024    4E75   RTS                                         1.    uS
+012     010041      15             supvr data write lb                1.    uS
+013     001026    4E56   LINK    A6,#missing operand, prefetch?      1.    uS
+014     0051F8    0000             supvr data read wd                 1.    uS
+015     0051FA    1244             supvr data read wd                 1.    uS

STATUS: M68000--Running in monitor     Trace complete                _____  0:59


   run     trace     step     display   modify    break     end     ---ETC---
```

Figure 11-6. Internal Analysis Trace of Assembly Language Module

RECEIVING TRIGGER ENABLE FROM ANOTHER ANALYSIS MODULE - EXAMPLE

The following measurement example illustrates how to execute an IMB measurement where the software analyzer is triggered by another analysis module. Using the example programs from the preceding section, the following example shows how to trigger the software analyzer from the HP 64302A Internal Analyzer. In this example, the internal analyzer traces the assembly language module **INIT_ACIA**. On completion of module execution, the software analyzer is triggered, showing the statements executed upon return to the Pascal procedure.

Setting Up the Emulator

To set up the emulator to drive the trigger enable signal, enter the emulator and execute the ***modify configuration*** command as in the previous example. This time, in response to the question " **Trigger enable?**, answer ***drive***.

Set up emulator run control with the following commands:

```
specify run from PASCAL_MAIN

specify trace before RINIT_ACIA
```

Setting Up the Software Analyzer

End out of the emulator and enter the software analyzer:

```
end

sw_anl_6
```

To set up the software analyzer to be triggered by the HP 64302A Internal Analyzer, modify the software analyzer setup configuration as follows:

```
setup measurement_disable off

setup measurement_enable on any_state

setup trigger_enable received measurement_enable
```

These commands turn off the measurement_disable function and gives control of the software analyzer's measurement enable function to the IMB.

Executing the IMB Measurement

Entering the execute command causes measurement execution, resulting in the measurement displays shown in figures 11-7 and 11-8. In figure 11-7, we see that the HP 64302A traced assembly language execution through the end of INIT_ACIA. In figure 11-8, the software analyzer trace shows execution of Pascal statements beginning with line 146, the first statement after the call to **INIT_ACIA**.

```
Trace:    mnemonic                        break: none        count:
 line#   address opc/data mnemonic opcode or status                time, relative
-015     00123E    5273              supvr pgm  read wd              <1.    uS
-014     001240    4EBA   JSR      0001014H                          1.    uS
-013     005273      20              supvr data write lb              1.    uS
-012     001242    FDD2              supvr pgm  read wd              1.    uS
-011     001014    13FC   MOVE.B   #043H,0010041H                   1.    uS
-010     0051F8    0000              supvr data write wd              1.    uS
-009     0051FA    1244              supvr data write wd              1.    uS
-008     001016    0043              supvr pgm  read wd              <1.    uS
-007     001018    0001              supvr pgm  read wd               1.    uS
-006     00101A    0041              supvr pgm  read wd               1.    uS
-005     00101C    13FC   MOVE.B   #015H,0010041H                   1.    uS
-004     010041      43              supvr data write lb              1.    uS
-003     00101E    0015              supvr pgm  read wd              <1.    uS
-002     001020    0001              supvr pgm  read wd               1.    uS
-001     001022    0041              supvr pgm  read wd               1.    uS
 before  001024    4E75   RTS                                        1.    uS

STATUS: M68000--Running in monitor     Trace complete                _____  0:59


  run     trace     step     display   modify    break     end     ---ETC---
```

Figure 11-7. Internal Analyzer Trace of INIT_ACIA

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Source                                    Symbol       Value      Stat   Time-rel

 146 HOWLONG := 220;   {initialize l*                                      0.0 uS
 147 TEN := 10;                           HOWLONG          220 write     1.5 uS
 149 FOR I := 1 TO WINDOW_SIZE DO  {* TEN                   10 write     2.2 uS
 150 WINDOW[I]:=' ';                      I                   1 write     2.1 uS
 150      "                               I                   1 read      2.1 uS
 150      "                               WINDOW[1]         " " write     4.4 uS
 150      "                               I                   1 read      1.5 uS
 150      "                               I                   1 read      3.0 uS
 152 BLANK_ARRAY;      {initialize t* I                       2 write     0.5 uS
 150 WINDOW[I]:=' ';                      I                   2 read      1.4 uS
 150      "                               WINDOW[2]         " " write     5.8 uS
 150      "                               I                   2 read      1.5 uS
 150      "                               I                   2 read      3.0 uS
 152 BLANK_ARRAY;      {initialize t* I                       3 write     0.5 uS

STATUS: Awaiting Command                                      52    1:37


   run     setup   db check   display   modify    show     execute   ---ETC---
```

Figure 11-8. Software Analyzer Trace of Statements Following Call to INIT_ACIA"

NOTES

Chapter 12

SELECTING AND FORMATTING THE MEASUREMENT DISPLAY

OVERVIEW

This chapter provides the following information about the measurement display:

- How to view data on the display
- Description of display fields
- How to interpret the displayed information
- Display command syntax

GENERAL INFOMATION

This chapter describes how to select the data displayed in the measurement display and how to format the information to increase its usability. The measurement results are automatically displayed upon completion of a measurement. When the measurement results are displayed, you can format the results on the screen using the ***display*** command. The ***display*** command allows you to select which data fields are displayed, the width of the fields, and the numeric base in which symbol values are displayed.

This chapter also describes the ***show*** and ***copy*** commands. The ***show*** allows you to select the setup display, measurement display, or the source program for display. The ***copy*** command allows you to copy the display, setup, or measurement results to the system printer or a listing file on the system disc.

If you have any difficulties or problems when using the software analyzer, see appendix E, Resolving Measurement Problems, for possible solutions.

VIEWING DATA ON THE DISPLAY

The [ROLL UP], [ROLL DOWN], [NEXT PAGE], and [PREV PAGE] keys allow the user to scroll through the trace listing line-by-line or in page increments. The left and right cursor ([←] and [→]) keys used with the [SHIFT] key allow the user to move the display left or right on the screen. Pressing [RETURN] with no command left justifies the display. In addition, the user can enter an integer value (>= 0) that specifies the state in the trace data buffer to be centered on the display (highlighted in inverse video). If the state is not the first state in a display line, the software analyzer will attempt to align the display to the first state in the line. If the display seems incorrect, try display positioning to return it to normal.

DISPLAY FIELDS

The measurement display can consist of up to eight different information fields; ***source***, ***source_path***, ***symbol***, ***symbol_path***, ***value***, ***status***, and ***count***. The following paragraphs describe the information fields.

Source Field

The ***source*** field is made up of source file statements corresponding to line numbers contained in the asmb_sym file created when the file was compiled. The software analyzer compares addresses in the trace record with addresses associated with symbols in the analyzer data base. If the analyzer detects an address corresponding to a line number symbol in the data base, it extracts the source statement with its line number from the source file and stores it in the display buffer for display in the trace list. If a source line is not found, "????" is displayed in place of the line number and a message explaining the reason is displayed in the ***source*** field.

Source Path Field

The ***source_path*** field contains the file name and userid of the source file from which the source statement for the current display line was extracted.

Symbol Field

The ***symbol*** field contains the symbols traced by the software analyzer. In the ***trace modules*** measurement, the symbols are the names of the modules traced. In the trace data_flow measurement, the ***symbol*** field consists of a module name with the ***symbol*** field of subsequent lines containing the names of the parameters and variables being traced on entry to or exit from the named module. In the trace variables and trace statements measurements, the ***symbol*** field consists of variable and parameter names. In the time modules measurement, the ***symbol*** field contains the names of the modules being measured. The symbol field is not displayed in the count statements and break measurements.

Symbol Path Field

The ***symbol_path*** field shows the path in which the symbol is defined. For modules, the symbol path contains a file name and userid. For variables and parameters, the symbol path may be a module name and file name with userid, or a file name and userid, depending upon the level at which the symbol is defined.

Value Field

The ***value*** field contains the values of the variables and parameters traced in the trace data_flow, trace statements, or trace variables measurements. This field is not valid in the trace modules measurement and cannot be displayed in the trace list display for that measurement. The default ***value*** field shows the data values in the data type notation specified in the Pascal or C source program, i.e., integers are displayed as integer values, real numbers as mantissa and exponent portions, boolean values as true or false, etc.

You can display values in ascii, binary, octal, decimal, hexadecimal, or the default format using the ***display base*** command. Variables with data structures greater than eight bytes in size can only be displayed in hexadecimal or default format. If any other base is selected, the variable will be displayed in default format. In ASCII display format, non-printable characters are displayed as hexadecimal values and any byte values greater than 7F (hexadecimal) are displayed as hexadecimal values followed by a question mark (?). Illegal values, partial values, and pad bytes are always displayed as hexadecimal values followed by a question mark (?). Changing the base causes the ***value*** field in all measurement modes to be changed.

When modifying variables such as sets, it is useful to display values in binary or hexadecimal to determine the value to use in the ***modify*** <*VAR*> command.

Status Field

The ***status*** field indicates whether the traced operation is an entry or exit from a module if the traced symbol is a module name, or the traced operation is a read from or write to memory if the traced symbol is a variable or parameter.

Count Field

The information displayed in the ***count*** field is dependent on the type of measurement being made. The count fields for different measurements are described in the following paragraphs.

TRACE MEASUREMENTS In the ***trace modules***, ***trace statements***, and ***trace variables*** measurements, the count field contains the number of bus states or the time measured from the first state of the current display line to the first state of the immediately preceding line (relative mode) or to the first state in the trace memory (absolute mode). If the counter is set ***to_count_states***, the field will be labeled "Count-rel" or "Count-abs". If the counter is set ***to_count_time***, the field is labeled "Time-rel" or "Time-abs". Absolute or relative display mode is selected with the ***display*** command explained in this chapter. The count field is not available in the ***trace data_flow*** measurement.

COUNT STATEMENTS. In the ***count_statements*** measurement display, the count field displays the number of times the associated source program line was executed. The ***counter*** parameter is automatically defaulted to ***to_count_states*** when a ***count_statements*** measurement is specified. The count field is labeled "count-abs" on the display.

TIME MODULES. In the time modules measurement display, the field displays the maximum, minimum, and mean execution times of the specified modules and the number of times the module was traced. The ***counter*** parameter is automatically set to ***to_count_time*** when a ***time_modules*** measurement is specified.

INTERPRETING THE DISPLAY

The following examples describe conventions and features of a trace list when displayed or copied to a list file. The examples were generated using the measurement command ***trace variables B file EXAMPLE***. Most of the discussion is applicable to displays generated by any software analyzer measurement. Part of the list file generated when file EXAMPLE was compiled is shown in figure 12-1. The TYPE and VAR definitions in the program must be understood before the sample displays can be understood.

```
  ...
  10 00000000  1  TYPE
  11 00000000  1  SHAPE          =(LINE,TRIANGLE,SQUARE,PENTAGON,HEXAGON,
  12 00000000  1                    HETAGON, OCTAGON, CIRCLE);
  13 00000000  1  POLYGON        =TRIANGLE..OCTAGON;
  14 00000000  1  SHAPE_SET      =SET OF SHAPE;
  15 00000000  1  POLY_ARRAY     =ARRAY[-2..0] OF POLYGON;
****WARNING ??                                    ^508
  16 00000000  1  PTR            =EC;
  17 00000000  1  REC            =RECORD
  18 00000000  2    FOO          :PTR;
  19 00000000  2    TIME         :REAL;
  20 00000000  2    FIGURE1      :SHAPE;
  21 00000000  2    FIGURE2      :SHAPE;
  22 00000000  2    P            :POLY_ARRAY;
  23 00000000  2    CHAR1        :CHAR;
  24 00000000  2    FLAG         :BOOLEAN;
  25 00000000  2    SETT         :SHAPE_SET;
  26 00000000  2    N            :UNSIGNED_16;
  27 00000000  2    CASE M       :SIGNED_16 OF
  28 00000000  2    2:(VARIANT2  :SIGNED_16);
  29 00000000  2    1:(VARIANT1  :SIGNED_8);
  30 00000000  1    END;
  31 00000000  1
  32 00000000  1  VAR
  33 00000000  1  A,B   :REC;
  ...
508: Warning: field or entry alignment; record or array comparisons may not work
```

Figure 12-1. Compiler Listing File For Program EXAMPLE

Current Line

In the trace list shown in figure 12-2, the underscored line in the center of the display is the current line. The number "912" on the status line is the first acquisition state for the current line. The number of states required for one line of display is variable.

```
 64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Symbol              Value          Stat  Source              Source path
B.FOO               000012EF6H     write   53 B:=A;          NT2:TEST
B.TIME              1.20000E-4     write   53 B:=A;          NT2:TEST
B.FIGURE1               CIRCLE     write   53 B:=A;          NT2:TEST
B.FIGURE2            PENTAGON      write   53 B:=A;          NT2:TEST
B.P[-2]                SQUARE     write   53 B:=A;          NT2:TEST
B.P[-1]              TRIANGLE     write   53 B:=A;          NT2:TEST
B.P[0]               OCTAGON      write   53 B:=A;          NT2:TEST
B.P[01H?]                72H      write   53 B:=A;          NT2:TEST
B.CHAR1                  "("      write   53 B:=A;          NT2:TEST
B.FLAG                  TRUE      write   53 B:=A;          NT2:TEST
B.SETT           [LINE,SQUAR*     write   53 B:=A;          NT2:TEST
B.N                      100      write   53 B:=A;          NT2:TEST
B.M                        1      write   53 B:=A;          NT2:TEST
B.VARIANT1                 0      write   53 B:=A;          NT2:TEST
B.+0017H?                00H      write   53 B:=A;          NT2:TEST

Status: Awaiting command                                    912   14:49

   run     setup    db check   display    modify     show     execute   ---ETC---
```

Figure 12-2. Sample Display Showing How Pad Bytes, Variant Records, and Field Widths Are Displayed

Displaying Pad Bytes

Line 53 in source file EXAMPLE moves variable A in its entirety to variable B. The ***symbol*** field in the eighth line of the display contains a question mark ("?"). A question mark in the ***symbol*** field indicates that a complete symbolic name does not exist for the value. The compiler warning in the type definition of POLY_ARRAY indicates that the physical size of a variable of type POLY_ARRAY is larger than the logical size. This is done to ensure that the subsequent field begins on an even byte boundary. The result is a "hole" or "pad byte" in variable B which has no symbolic name. The analyzer recognizes that the byte is physically part of array P but that an index of 1 is not valid. A question mark will always be displayed when the analyzer traces a pad byte corresponding to a "508" warning from the compiler.

Displaying Variant Records

The question mark in the 15th line is slightly different. In Pascal, a record is physically large enough to accommodate its largest variant. In C, a structure is physically large enough to accomodate its largest union. In this case, the record B must be large enough to accomodate VARIANT2 which is two bytes long. Unless a specific variant is requested in the setup command, the analyzer defaults to displaying a record with the first variant (C language) or the last variant (Pascal language). The byte which is offset 17H bytes from the first byte of B is defined with respect to VARIANT2 but not with respect to VARIANT1. The analyzer always displays undefined bytes as +nnH?, where nn is a byte offset from the first byte of the record.

Field and Display Width

The "*" in the *value* field of the 11th line of the display indicates that the field is not large enough to display the entire value of B.SETT. The size of the field may be increased by using the display command to specify a larger width. The asterisk may appear as the last character in any field, indicating that there is additional information that is not being displayed. The display may be reformatted with a greater width for the field containing the asterisk. Each display field can have a maximum width of 132 characters. The display has a maximum width of 220 characters. The display window may be moved for viewing by using the SHIFT key with the left or right arrow.

Illegal Values

Figure 12-3 shows another display resulting from the *trace variables B file EXAMPLE* measurement. Question marks in the *value* field generally indicate that the value of a symbol is illegal with respect to the symbol type, i.e., it is out of range. The "07H?" in the seventh line is out of range because the value 7 (CIRCLE) is not a legal value for a POLYGON. In the ninth line, "81H" is not a legal ASCII character. In the 10th line, "0CH" is not a valid boolean value. The "0AH" in the 11th line indicates that the bit for the 10th element of B.SETT was set but there is no 10th element in the definition of the set (first element = element 0).

```
64340 Software Analyzer: Slot 6   with   em68000  Emulator: Slot 4

Symbol              Value          Stat  Source              Source path
B.FOO          ?????2EF6H write   53 B:=A;          NT2:TEST
B.TIME          -Infinity write   53 B:=A;          NT2:TEST
B.FIGURE1          CIRCLE write   53 B:=A;          NT2:TEST
B.FIGURE2       PENTAGON write   53 B:=A;          NT2:TEST
B.P[-2]          SQUARE write   53 B:=A;          NT2:TEST
B.P[-1]         OCTAGON write   53 B:=A;          NT2:TEST
B.P[0]             07H? write   53 B:=A;          NT2:TEST
B.P[01H?]           72H write   53 B:=A;          NT2:TEST
B.CHAR1            81H? write   53 B:=A;          NT2:TEST
B.FLAG             0CH? write   53 B:=A;          NT2:TEST
B.SETT      [LINE,0AH?] write   53 B:=A;          NT2:TEST
B.N               100 write   53 B:=A;          NT2:TEST
B.M                 2 write   53 B:=A;          NT2:TEST
B.VARIANT1          0 write   53 B:=A;          NT2:TEST
B.+0017H?         00H write   53 B:=A;          NT2:TEST

Status: Awaiting command                                  912   14:49


  run    setup   db check   display   modify    show    execute   ---ETC---
```

Figure 12-3. Example Display Showing Illegal Values, Special Values, and Incomplete Access to Values.

Special Values

B.TIME in the second line of the display has a value of "-infinity". Real numbers may also have the special values "+infinity" and "not a number".

Incomplete Access To Variables

The four question marks in the value of B.FOO indicate that the analyzer has seen an incomplete access to the variable and has acquired only part of its value. In this example, the partial value is due to the analyzer beginning data capture in the middle of the two bus cycles that wrote to B.FOO. The asynchronous nature of the measurement resulted in the first write cycle not being captured.

Partial values also occur when the executing code accesses one traced variable and then a second traced variable without having accessed all of the first variable. A common example is when the measurement is tracing two 32-bit integers where one is assigned to the other. If the code does the assignment with word moves, partial values result from the second variable being written to before all of the first variable is read. Similarly, partial values are common when tracing two structured variables where one is assigned to the other.

Errors may occur in trace statements measurements when partial values are displayed adjacent to a complete value of the same variable. The software analyzer groups bus cycles together into what it considers to be one logical access. This grouping may be incorrect when a access is adjacent to complete accesses. The user may alter the grouping by specifying a new integer position to be shown on the center line of the display. The software analyzer uses this integer position as the initial condition in its grouping algorithm. This will alter the manner in which the bus cycles are grouped.

STATE NUMBER

When the measurement is displayed, you may enter an integer value from 0 to 9999 specifying the position in the measurement buffer to be displayed and centered on the screen.

```
344 RETURN
```

trace display

The *display* command is used to specify the format of the information displayed in the trace listing. This command allows the user to select which fields to display, the sequence in which the fields are displayed, field width, and other parameters described below.

Command Syntax

The *display* command syntax is shown in figure 12-4.

Parameters

The following paragraphs describe the parameters used in the *display* command.

absolute — *absolute* specifies that counts or times be displayed as an absolute count from the beginning of the trace to the current line.

ascii — *ascii* is used with the *display base* command to display values in the value field as ASCII characters.

base — *base* is used to specify the base that the value field is to be displayed in. The choices are default (native type assigned in the source program), ascii, binary, octal, decimal, or hex.

binary — *binary* is used with the *display base* command to display values in the measurement display value field as binary numbers.

count — *count* specifies that the count/time field be displayed in the measurement display. The count/time field may be displayed in relative, absolute, and statistics mode, depending upon the type of measurement made.

decimal — *decimal* is used with the *display base* command to display values in the measurement display value field as decimal numbers.

default — *display default* specifies that the measurement display be displayed in the default format. *display base default* specifies that the value field be displayed in the default base, i.e., in the data type assigned in the source program.

hex — *hex* is used with the *display base* command to display values in the measurement display value field as hexadecimal numbers.

modify — *modify* is used with the *display* command to modify the current display definition. The *display modify* command recalls the current display definition to the command line for editing, eliminating the need to re-enter an entire display command.

octal — *octal* is used with the *display base* command to display values in the measurement display value field as octal numbers.

display (Cont'd)

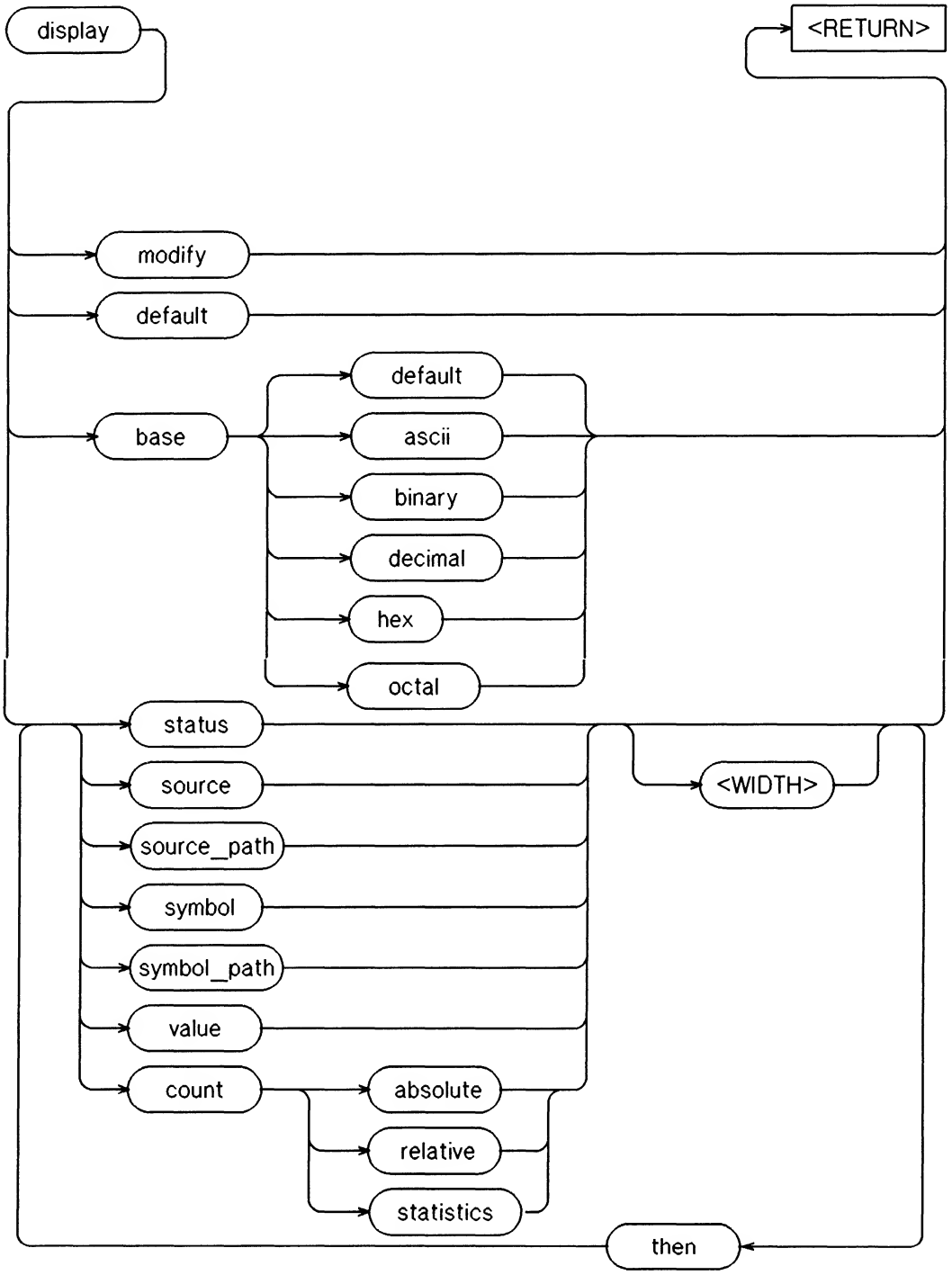


Figure 12-4. Display Command Syntax Diagram

display (Cont'd)

relative	***relative*** specifies that counts or times on a line should be displayed relative to the count or time on the preceding line of the measurement display.
source	***source*** specifies that the ***source*** field is to be displayed in the trace list.
source_path	***source_path*** specifies that a field is to be displayed showing the source file name that the source statement was extracted from.
statistics	***statistics*** specifies that the statistics field (minimum, maximum, mean, and count) be displayed on the measurement display. This field is valid only for ***time modules*** measurements.
status	***status*** specifies that the ***status*** field is to be displayed in the trace list.
symbol	***symbol*** specifies that the ***symbol*** field is to be displayed in the trace list.
symbol_path	***symbol_path*** specifies that a field is to be displayed showing the path in which the symbol is defined. For modules, the ***source_path*** contains a file name. For variables and parameters, the ***source_path*** may be a file or a module and file, depending upon the level at which the symbol is defined.
then	***then*** is used as a delimiter to separate field definitions when more than one field is specified in the display command line.
value	***value*** specifies that the ***value*** field is to be displayed in the trace list. The values are displayed in the notation of the data type declared in the source program.
<WIDTH>	<WIDTH> is a prompt for the user to define the width of a displayed field in columns. Width must be an integer value from zero to 132 columns.

Display Command Examples

The following examples illustrate how to use the ***display*** command to format the trace list display.

```
display default
display source 40 then symbol 10 then value
display symbol 10 then value 10 then source then
   source_path
```

Chapter 13

CONFIGURING THE ANALYZER

OVERVIEW

This chapter describes the following three methods of configuring the software analyzer:

- Getting the measurement configuration last used
- Getting a measurement from a configuration file
- Configuring a measurement with a command file

GENERAL INFOMATION

This chapter explains how to configure the software analyzer. The analyzer can be configured manually each time it is used or it can be configured automatically. There are three methods that you can use to load the analyzer configuration automatically when you first enter the analyzer: (1) you can use the *options continue* feature to recall the measurement setup you used to perform the last tests, (2) you can use any measurement setup that has been previously stored in a configuration file, or (3) you can use a measurement system command file. Each of these methods are discussed in this chapter.

NOTE

The software analyzer recomputes the addresses of symbols each time the analyzer is configured from a configuration file. If any programs referenced in a configuration file are changed such that symbol names are modified or deleted, the configuration file may no longer be valid. Errors will occur if symbols are no longer present or have been modified.

GETTING THE MEASUREMENT CONFIGURATION LAST USED

When you are running a measurement session and you press the *end* softkey the first time, the analyzer keeps the current measurement configuration in on-board RAM and moves you to the measurement system software level. You can enter the other analysis functions available at this level if you wish. To return to the software analyzer softkey level, press the *sw_anl_N* softkey, then the RETURN key.

Pressing the ***end*** softkey a second time brings you out of the measurement system level software to the system monitor level software. Here you can use the system monitor level softkey functions without disturbing the measurement setup you ended out of as long as you do not press the ***opt_test*** softkey at the monitor level. You can reenter the software analyzer measurement session with the last configuration used at any time by pressing the ***meas_sys*** and ***continue*** softkeys, and then the (RETURN) key.

NOTE

If you do not include the ***options continue*** statement in your command, your present measurement configuration will be purged.

The ***options continue*** function will not perform the function described above after the (RESET) key has been pressed twice, or after a power down or power fail, or after running performance verification.

If you have entered the same emulator used by the analyzer after ending the analysis session and modified the emulation setup (i.e. loaded a new file while in emulation), an attempt to reenter the software analyzer using the ***options continue*** statement may fail.

GETTING A MEASUREMENT CONFIGURATION FROM A CONFIGURATION FILE

The analyzer can store complete measurement configurations in disc memory so that you can keep a library of test setups and measurement data on hand for your measurement needs. You can then load a selected measurement configuration to suit your current need without having to build a new configuration for each measurement session. If you have a configuration file that is close to the configuration you need, you can load it and then modify it, saving time by eliminating the requirement to enter some basic parameters. The following paragraphs describe the procedures used to store and recover these measurement configurations. The syntax for saving or loading a configuration is shown in figure 13-1.

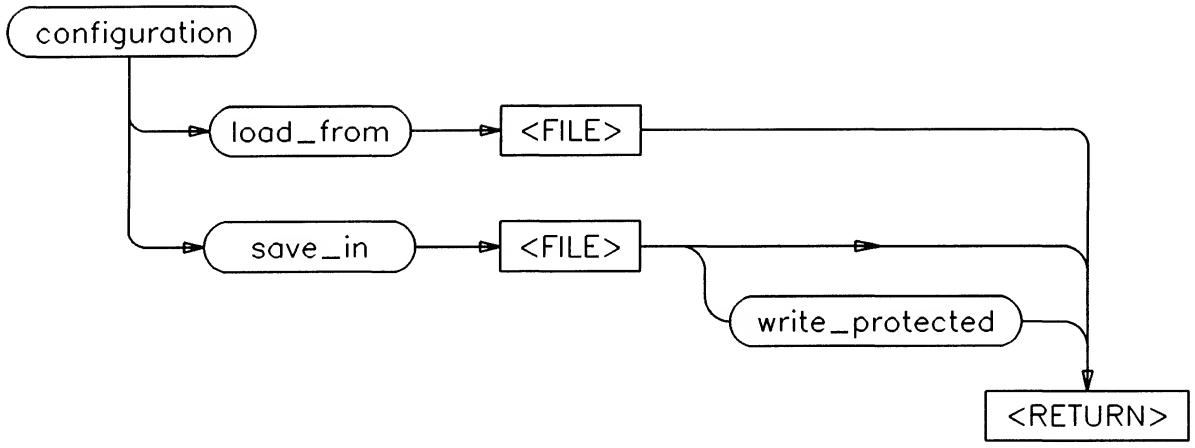


Figure 13-1. Configuration Syntax Diagram

Saving A Measurement Configuration

1. Set up any desired measurement configuration in your software analyzer. It is a good idea to set up a good basic configuration that can be stored, then loaded and used as a building block for other measurement configurations.

2. Press the *configure* and *save_in* softkeys, then type in an A in answer to the *<FILE>* softkey prompt. Now press the RETURN key. The analyzer will now save its present measurement configuration in the trace file you have just named A. The file will be stored under the current USERID.

3. Now you can change the setup any way you like. Your original measurement configuration will still be saved exactly as you stored it in file A. You can use this procedure to make as many configurations as you may require. These, in turn, can also be stored in configuration files for access at a later time.

If you used the *write_protected* option when you saved your configuration and you ever want to purge that file, you must return to the system monitor level software to accomplish the purge. To accomplish this press the *end* softkey, then the RETURN key (in that sequence) two times. This will return you to the system monitor level software. You can now purge the unwanted file.

Loading A Measurement Configuration

If you are starting a session (or are at the measurement system monitor level) and want to load a configuration you have previously stored in a file, proceed as follows: press the *sw_anl_N* softkey, type in the name of the configuration file you want to use, and press the RETURN key. You will gain access to the software analyzer and it will automatically search the disc and load the configuration you stored in the file you requested.

If you are operating the software analyzer in a measurement session and you want to load a configuration you have stored in a file without ending out of the analyzer, proceed as follows: press the *configure* and *load_from* softkeys, type in the name of the file you want to use, and press the RETURN key. This will cause the analyzer to purge the present measurement setup and load the configuration from the file you requested.

CONFIGURING A MEASUREMENT WITH A COMMAND FILE

The command file must contain the sequence of command lines required to create the setup from the monitor level. Using the parameter passing feature of command files will allow the emul_cmd, absolute, and source files required by the software analyzer to be specified in the command file. An example is shown below:

```
PARMS &CMD_FILE &ABS_FILE &SOURCE_FILE
measurement_system
sw_anl_6 &CMD_FILE
load &ABS_FILE
setup default_path &SOURCE_FILE
setup trace statements line 82 to 100
run at_execution from transfer_address
execute
wait measurement_complete
copy measurement to MEASURE append

Where:
    CMD_FILE is the emulation command file
    ABS_FILE is the linked absolute file to be traced
    SOURCE_FILE is the source file to be debugged
```

The command file can be run from the system monitor level. To run the command file, type in the file name and the file names for the emulation command file, absolute file, and source files to be used.

<CMDFILE> STEP2 NT1 NT1 RETURN

If the file names are not entered, the prompts:

Define parameter &CMD_FILE:
Define parameter &ABS_FILE:
Define parameter &SOURCE_FILE:

will appear on the command line.

Chapter 14

USING SUPPORT COMMANDS

OVERVIEW

This chapter contains the following information:

- System software conventions
- System utilities available from within the software analyzer
- Software analyzer utilities

GENERAL INFORMATION

This chapter describes the software conventions used to make keyboard entries for operating the software analyzer and how the analyzer directs the entries you make. Also described are the utility softkeys, the utility keyboard keys, and the prompt softkeys.

SYSTEM SOFTWARE CONVENTIONS

This section contains information concerning the system software as it relates to any of the subsystems installed in a particular mainframe.

User Identification

The user identification (userid) command is the means of identifying yourself to the HP 64000 system software as a unique individual who will be using the system for your own analysis/development projects. Signing onto the system with your own userid immediately identifies which group of files the system is to work with.

The *userid* syntax is a string of up to six (6) alphanumeric characters which start with an upper case alpha character. If you select a userid with more than six characters, the system will recognize only the first six. If you do not select a userid, the default condition is a blank userid. A blank userid limits your ability to designate a file because if more than one file is given the same name, and that file is called up, the system will recognize the first one it sees (which may or may not be the one you want).

Directed Syntax

The system software causes a row of softkey labels to be displayed across the bottom line of the CRT display. These softkey labels identify the functions to be obtained by pressing corresponding keys in the row at the top of the keyboard. When you press one of the softkeys (selecting a parameter), the names of all the softkey labels change. The new softkey names offer selections that can be made to complete the command entry.

By directing the syntax of your entries, syntactical errors are virtually eliminated. The softkey label line always identifies appropriate entries to be made at any point during the process of formulating a command. The software analyzer softkeys always prompt the user with a *<RETURN>* softkey label when a valid command statement has been entered. If the softkey label line contains more labels than the *<RETURN>* softkey prompt, then the command statement may either continue or be terminated by pressing the RETURN key, as determined by the specific requirement of the command being formulated.

Entering Numeric Values

You can enter numbers into an analysis specification in any of the four standard number bases. Place the applicable letter symbol (B, O or Q, D, H) at the end of your number to define its base. Refer to the following examples:

1000B = 1000 binary
1000O or 1000Q = 1000 octal
1000H = 1000 hexadecimal
1000D or 1000 = 1000 decimal

Hexadecimal numbers beginning with a letter must be preceded with a numeric zero. For example:

3FAH, 0FFH, 0F44H (but not F44H)

NOTE

Decimal is assumed if no base is specified when a number is entered.

Entering Module/Variable Names

You can enter module and variable names into an analysis specification exactly as it appears in the source program with one exception. If the module or variable name is lower case and is identical to a software analyzer keyword, it must be specified in quotes, e.g., "entry". Otherwise the analyzer interprets the name as a keyword and generates an error message.

File Names

File names may consist of from one up to nine alphanumeric characters, starting with an upper case letter. Underscores (_) are also permissible. Alpha characters, after the first character, may be upper or lower case.

SYSTEM UTILITIES

Several system utilities and features can be used within the software analyzer. These features are described in the following paragraphs.

Command Files

A command file is a source file containing a sequence of commands as they would appear on the command line if entered manually from the softkeys or keyboard. A command file is used to create a particular measurement configuration programatically. A command file provides a self-documenting record of a measurement setup and allows easy editing and modification. By using the *wait* command described in this chapter, command files can be set up to perform some automated measurements which require no operator interaction.

A semicolon (;) is used in the command file to denote comments. The analyzer software will not read any material following a ";" in any line of a command file. It will start loading new instructions only after it finds the next carriage return.

In the example:

```
run  from  transfer_address; causes program execution to begin
```

only the command line text "*run from transfer_address*" will be acted on.

Logging Commands

The HP 64000 Logic Development System has the capability to log commands to a command file. This feature is especially useful for building command files that will carry out the entire measurement setup automatically. To log commands for a measurement setup session from the system monitor level software, press the *log* and *to* softkeys, type in the name of the file you want to use, and press the RETURN key. From this point until you are once again back in the system monitor software and press the *log* and *off* softkeys, and the RETURN key, all of the valid commands you entered are logged into the log file. You may then conduct a software analysis session which will build a command file for later use or for modification.

Recall Key

The RECALL key will cause the analysis module to return the preceding valid command line to the screen. The analysis module has a command line memory which the RECALL key accesses. Each time you press the RECALL key, the analyzer steps one execution further back into its memory of command lines.

Tab Key

The TAB key is used to move the cursor rapidly through the command line on screen. This key is useful when you are making modifications to long specifications. By pressing TAB, you step the cursor from entry to entry forward through the specification on the command line. By pressing the SHIFT key and then the TAB key, you step the cursor backwards through the specification.

Insert Char And Delete Char Keys

The (INSERT CHAR) and (DELETE CHAR) keys are used to edit the content of the command line. The (INSERT CHAR) key will open a space before the present position of the cursor so that you can add entries in the command line. The remainder of the line will automatically shift to the right with each new entry that you make. The (INSERT CHAR) key function will remain in effect until it is pressed again or until any other utility key is pressed (except (←), (→), or (CAPS LOCK)). The (DELETE CHAR) key is used to eliminate entries from the the command line without losing the entire specification. When you press the (DELETE CHAR) key, the entry directly over the cursor will be eliminated and the remainder of the specification will shift left. Holding the (DELETE CHAR) key down will cause multiple character deletions as characters are shifted left, over the cursor position.

Prompt Softkeys

Any softkey name enclosed in angle-brackets "<>" is a prompt for the operator. If you press a prompt softkey, the STATUS line of the display will explain the meaning of the prompt. The software analyzer softkey label prompts and their corresponding status line prompt messages are given in appendix B.

SOFTWARE ANALYZER UTILITIES

The software analyzer utility commands are ***copy***, ***end***, ***execute***, ***halt***, ***setup modify***, ***show***, and ***wait***. These softkeys allow the user to execute a measurement, copy and display information concerning the current session, and to end the analysis session without losing the current measurement specification. In addition they allow the user to halt a measurement in progress or to disable commands while a measurement is in progress. These softkey commands are described in the following paragraphs.

copy

The *copy* command is used to copy the current display, setup, or all or part of the measurement data (trace list) to a listing file or to the system printer. The information may be appended to an existing listing file.

Command Syntax

The *copy* command syntax is shown in figure 14-1.

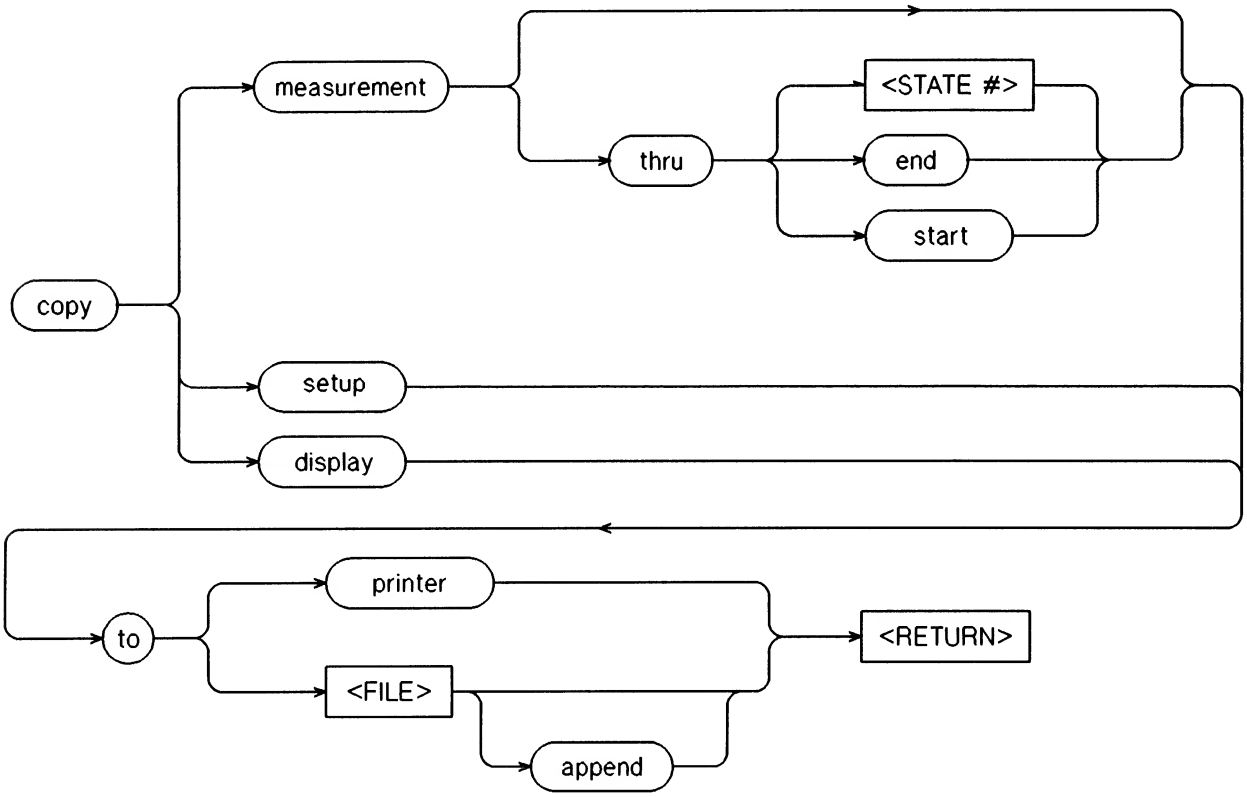


Figure 14-1. Copy Command Syntax Diagram

Parameters

The following definitions describe the parameters used in the *copy* command.

append — *append* specifies that the display, setup or measurement being copied be appended to the end of the listing file.

copy (Cont'd)

display	***display*** specifies that an image of the current display be copied to the specified file or system printer.
<FILE>	<FILE> is a prompt for the user to enter the file name of the listing file that the display or trace list is to be copied to.
measurement	***measurement*** specifies that the measurement trace list or specified portion of the trace list be copied to the listing file or system printer.
printer	***printer*** specifies that the display or trace list be copied to the system printer.
setup	***setup*** specifies that the measurement setup information be copied to the listing file or system printer.
<STATE #>	<STATE #> is a prompt to the user that a integer specifying a location in the measurement data buffer may be entered, specifying the measurement data from the current line on the display (displayed in inverse video) to the specified state be copied.
thru	***thru*** is used to specify which portion of the trace listing is to be copied. ***thru end*** specifies the first line of the current display through the end of the trace list. ***thru start*** specifies the start of the trace list through the last line of the current display. ***thru*** <STATE#> specifies a record position in the trace measurement buffer. The trace list from the record position to the top or bottom of the current display will be copied, depending on whether the specified record position occurs after or before the currently displayed data. The minimum data that can be copied is the current display.

Copy Command Examples

The following examples illustrate how to use the ***copy*** command.

```
copy display to printer
copy setup to SETUP2
copy measurement thru 50 to TRACE1 append
copy measurement thru start to printer
```

When a ***copy*** command is executed, the acquisition state for each line of the measurement display or measurement listing is copied along with the line.

end

When you press the *end* softkey, the HP 64000 station exits the software analyzer and returns to the measurement system level software. Pressing the *end* softkey a second time causes the software to enter the system monitor level of software. You can make use of the system monitor level softkey functions without disturbing the measurement you ended from as long as you do not press the *opt_test* softkey at the monitor level.

If you pressed the *end* softkey only once, the instrument is at the measurement system software level. From here you can reenter the analysis module by pressing the *sw_anl_N* softkey and the RETURN key. If you pressed the *end* softkey twice the instrument is at the system monitor software level. From here you can reenter the analysis module by pressing the *meas_sys* and *continue* softkeys and the RETURN key.

Command Syntax

The *end* command syntax is shown in figure 14-2.

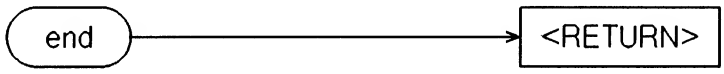


Figure 14-2. End Command Syntax Diagram

execute

The *execute* softkey causes the analyzer to initiate a measurement based on the parameters defined in the measurement setup specification. Pressing the *execute* softkey, then the RETURN key causes the analyzer to search for and acquire data as specified in the trace setup specification. While the measurement is in progress, the STATUS line displays "*Executing real-time ...*". When the measurement is complete, the STATUS line displays " *unloading acquisiton memory (count = nnnn)*" while the acquired data is being read to the measurement file. "*formatting display*" is displayed while the data is being formatted for display. When formatting is completed, the measurement will appear on the screen. Whenever the *execute* command is given, the last measurement file is deleted.

Command Syntax

The *execute* command syntax is shown in figure 14-3.

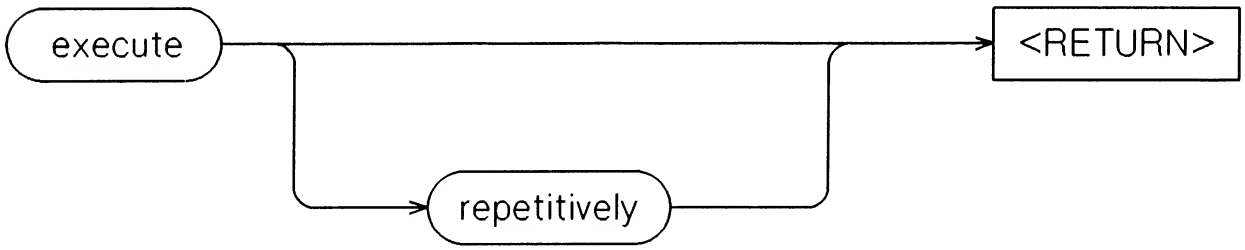


Figure 14-3. Execute Command Syntax Diagram

Parameters

The *execute* command has one optional parameter, *repetitive*.

repeat	*repetitive* specifies that the measurement be executed repetitively until a *halt* command is given.

halt

The ***halt*** softkey is used to (1) halt execution of the current measurement or (2) halt unloading of the acquisition memory. The ***halt*** command is executed by pressing the ***halt*** softkey, then the RETURN key.

Termination of a measurement by halting does not cause measurement completion as defined for the ***setup break on measurement_complete*** and ***wait measurement_complete*** commands.

Command Syntax

The ***halt*** command syntax is shown in figure 14-4.

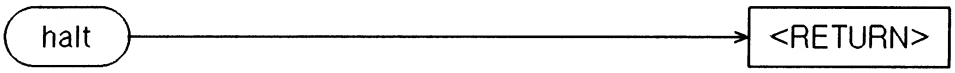


Figure 14-4. Halt Command Syntax Diagram

setup modify

The *setup modify* command recalls the last measurement setup, measurement enable, or measurement disable to the display command field for editing. This enables you to edit the command without having to retype the entire command.

Command Syntax

The *setup modify* command syntax is shown in figure 14-5.

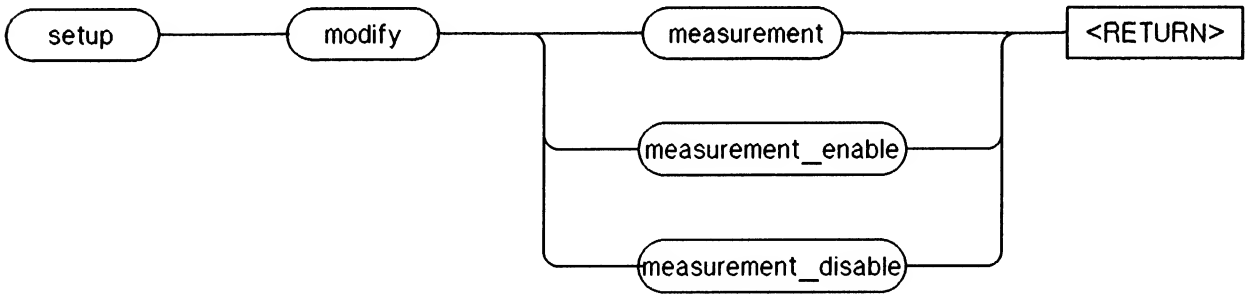


Figure 14-5. Setup Modify Command Syntax Diagram

Parameters

The following definitions describe the parameters used in the *setup modify* command.

measurement	*measurement* specifies that the last setup measurement command be recalled to the display field for editing.
measurement_enable	*measurement_enable* specifies that the last setup measurement enable command be recalled to the display field for editing.
measurement_disable	*measurement_disable* specifies that the last setup measurement disable command be recalled to the display field for editing.

show

The *show* command is used to select the measurement, setup, or a source program for display.

Command Syntax

The syntax for the *show* command is shown in figure 14-6.

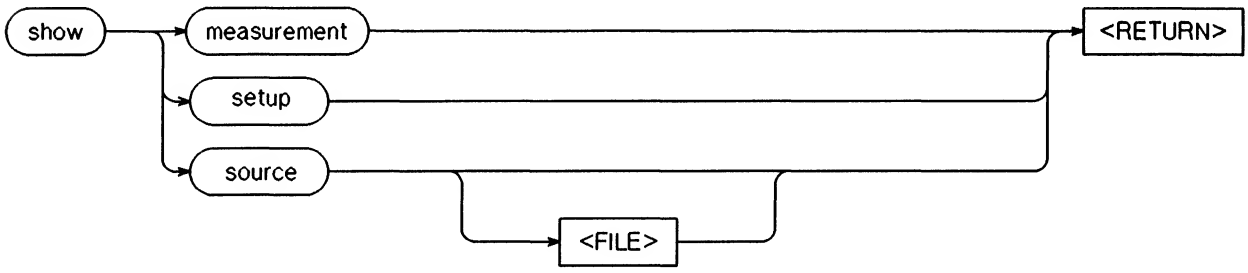


Figure 14-6. Show Command Syntax Diagram

Parameters

The following definitions describe the parameters used in the *show* command.

<FILE>	<FILE> prompts the user to enter the name of the source file to be displayed. If no file name is entered, the default path file is displayed.
measurement	*measurement* is specified in the *show* command to display the current measurement results.
setup	*setup* is specified in the *show* command to display the software analyzer setup.
source	*source* is used in the *show* command to display a source file. If no source file is entered in the command statement, the default path file is displayed.

Show Command Examples

The following command examples illustrate how to use the *show* command.

show setup
show measurement
show source BUB_SORT

wait

The *wait* command causes the software analyzer to disable all software analzyer commands until the wait condition is cleared. *wait* enables the user to create command files that can execute repetitive measurements, storing the measurement results between measurement executions. This command provides the capability to automatically make measurements, unattended by the user, with the results stored in listing files for future analysis.

The *wait measurement_complete* command causes the software analyzer to disable all software analyzer commands until the measurement is completed. When used in a command file, the *wait measurement_complete* command suspends execution of the command file until the measurement is completed.

The *wait <SECONDS>* enables you to execute a wait from one second to 65535 seconds (approximately 18.2 hours) in duration. If a larger value is entered, the analyzer truncates the value to its 16 least significant bits, e.g., 65537 seconds would be truncated to one second.

Pressing the RESET key when a wait condition is enabled will stop execution of the command file. Pressing any other key will clear the wait.

Command Syntax

The *wait* command syntax is shown in figure 14-7.

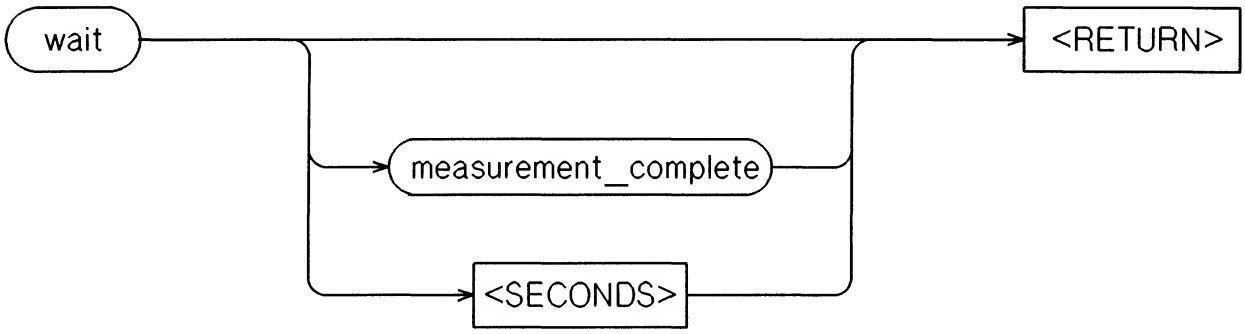


Figure 14-7. Wait Command Syntax Diagram

wait (Cont'd)

Parameters

The following definitions describe the parameters used in the *wait* command.

measurement_complete — *measurement_complete* causes suspension of command file execution until the current measurement is completed.

<SECONDS> — <SECONDS> prompts the user to specify the number of seconds to wait before again accepting commands. A value from 0 to 2^{16}-1 (65535) seconds may be specified.

Wait Command Examples

The following command examples illustrate how to use the *wait* command.

wait
wait measurement_complete
wait 120

NOTES

Chapter 15

SYMBOLS AND DATA TYPES

OVERVIEW

This chapter contains the following information about symbols and data types supported by the software analyzer:

- Supported symbol classifications (static and dynamic)
- Supported data types

GENERAL INFOMATION

This chapter describes the storage classes and data types that the software analyzer recognizes. Symbols are categorized by whether the location of the symbol is known at link time (static) or whether the location changes during run time (dynamic). The scalar and structured data types that can be symbolically referenced by the software analyzer are described.

SYMBOL CLASSIFICATIONS

Static Symbols

LOCAL AND GLOBAL VARIABLES. Static variables (both local and global) are defined as those whose base locations are allocated at link time. The software analyzer expects that these locations will not change during run time.

PROGRAMS, MODULES, PROCEDURES, AND FUNCTIONS. A module is a set of program statements that can be invoked (or referred to) by name. In Pascal, "module" may refer to the main program of a file, or to individual procedures or functions within a program. In C, "module" can refer only to functions. The key elements of a module, as required by the analyzer, are (1) the module must be a contiguous segment of code with a single entry point and a single exit point, and (2) all the code for the module must fall within the range of the entry and exit points. The entry point is defined in terms of the assembly code which is generated after compilation of the high level language module. It is the first executable instruction of the module (this includes any compiler overhead which may have to be done before the assembly code performing the actual module operations begins). The address of the entry instruction becomes the lower boundary of the address range for the module. The exit point is defined as the last executable instruction of the code segment and its address becomes the upper boundary of the address range for the module.

LABELS. The entry point into a module's assembly code must have a label associated with it which must be the module name (e.g. MAIN). The exit point must also have a label associated with it which is identical in the first 14 characters to the entry point label except that an "R" is appended to the front of the label (e.g. RMAIN). It is these labels, found in the symbol files, that the analyzer keys on to perform a table lookup of the address range associated with the module, as well as its entry and exit points. The compilers follow these design rules but may, under certain conditions, create identical labels. These conditions are as follows:

1. procedures and functions in Pascal which are on different levels (i.e. nested procedures) may have identical names.

2. Due to the creation of the "R" or exit point labels, procedures and/or functions identical in the first 14 characters on any level will produce identical "R" labels.

The analyzer always keys on the first label it encounters that matches the specified label. Therefore, in order to avoid having the analyzer key on the wrong label, it is recommended that you always make your procedure and/or function names unique within the first 14 characters.

LINE NUMBERS. The software analyzer provides symbolic lookup of line numbers. These line numbers correspond to the line numbers found in the compiled listing file. The analyzer only accepts line numbers having executable code associated with them. If a line number has several instructions associated with it, the first instruction is the instruction associated with that line number. Lines that are intermixed in high level code, but contain only comments, do have executable code associated with them. These lines will be associated with the executable code immediately following them. Any comments that occur before the beginning and after the end of a module do not have executable code associated with them and do not exist for purposes of the software analyzer.

PATHS. A path consists of a module name and source file name that uniquely identifies a variable. Possible module names include function names in C programs, and procedure and function names in Pascal programs. The procedure or function name may be qualified by a file name. In Pascal, the main program path is defined by the file name.

Proc. The keyword *proc* is found in the commands *modify <VAR> proc ...*, *display <VAR> proc ...*, *setup trace variables <VAR> proc ...*, and *setup trace data_flow ...*. The keyword *proc* is used to specify an element of the variable's path, i.e., *proc* defines the procedure or function the variable belongs to and enables unique identification of the variable.

File. The keyword *file* is used to describe the file a variable belongs to. If only a file is given with no *proc* specified, the variable belongs to the outermost level by default. In Pascal, the outermost level is the program level.

Default Path. A default path may be set up before an actual measurement command is given. Then, if no path name is specified in the measurement command, the default path is used as the path definition for the measurement. If no default path is defined, the path must be specified in the measurement command.

Dynamic Symbols

LOCAL VARIABLES. Dynamically activated local variables are those that are assigned to the stack when the procedure is invoked and taken off when the procedure ends. The variable can be in different places at different times during run time.

REFERENCE PARAMETERS. A parameter that is passed by reference causes the address of the parameter to be passed. If a variable passed by reference is traced, the changes that occur to that variable will be seen within the subroutine the parameter was passed from. The variable will also be displayed with the name it acquired after being passed as a parameter.

VALUE PARAMETERS. Value parameters are not active on exit from procedures since they are treated in the same manner as local variables by the compiler. In C, all parameters except arrays are passed by value.

SYMBOLIC DATA TYPES

Many types of data can be symbolically referenced by the software analyzer. The following paragraphs describe the data types recognized by the software analyzer.

Intrinsic Data Types

Table 15-1 describes the intrinsic data types recognized by the software analyzer.

Table 15-1. Intrinsic Data Types

Scalar Data Types

Pascal	C	Description
BOOLEAN	Not applicable	An 8-bit value whose low-order bit represents the value TRUE (1) or FALSE (0).
BYTE	short	An 8-bit signed integer in the range -128 to +127.
SIGNED_8	short	An 8-bit signed integer in the range -128 to +127.
UNSIGNED_8	unsigned short	An 8-bit unsigned integer in the range 0 to 255.
SIGNED_16	int	A 16-bit signed integer in the range -32768 to +32767.
UNSIGNED_16	unsigned	A 16-bit unsigned integer in the range 0 to 65535.
SIGNED_32	long	A 32-bit signed integer in the range -2,147,483,648 to +2,147,483,647.

Table 15-1. Intrinsic Data Types (Cont'd)

Scalar Data Types (Cont'd)

Pascal	C	Description
UNSIGNED_32	unsigned long	A 32-bit unsigned integer in the range 0 to +4,294,967,295.
CHAR	char	An 8-bit value in the set of characters defined by the 8-bit ASCII character set.
INTEGER	long	A 32-bit signed integer in the range -2,147,483,648 to +2,147,483,647.
REAL	float	A 32-bit binary value representing a floating point number in IEEE simple precision format.
LONGREAL	double	A 64-bit binary value representing a floating-point number in IEEE double precision format.

User-Definable Data Types

Pascal	C	Description
SCALAR TYPE	enum	A type that defines an ordered set of values by enumerating the identifiers which denote these values.
SUBRANGE TYPE	Not Applicable	A type that is identified as a subrange of a previously defined ordinal type (char, byte, integer, or scalar) in which the smallest and largest values are user defined.

Structured Data Types

The following paragraphs describe the structured data types recognized by the software analyzer.

ARRAY. An array is a structure consisting of a number of components which are all of the same type (called the component type), in which the components (elements of the array) are accessed by index expressions. The array type definition specifies the component type and, in Pascal, the index type.

In Pascal, the component type may be of any type. Multidimensional arrays may be represented as "ARRAY OF ARRAY (OF ARRAY..)" with an arbitrary number of indices. The index type must be a simple type such as scalar or subrange type.

In C, the component type may also be of any type. Multiple dimensions may be specified by multiple brackets, i.e., [size] [size] [size]. The index type must always be integer. In C, an array passed as a parameter with undefined size cannot be traced with the software analyzer.

POINTER. A pointer is a variable that contains the address of a dynamic variable such that the dynamic variable can be accessed via the pointer variable. In C, arrays are not considered pointers for the purposes of this manual. Pointers can be traced in all software analyzer measurements that trace variables. Pointer expressions (the dynamic variable accessed via the pointer variable) can be traced only with the ***trace data_flow*** measurement. Pointer expressions can be displayed or modified using the ***modify*** and ***display*** commands.

SET (Not Applicable to C). A set is a structure defining the set of values that is the power set of its base type,(i.e., the set of all subsets of values of the base type). The base type must be a scalar or subrange type.

RECORD/STRUCTURE. A record (structure in C) is a data type consisting of a fixed number of components, called fields (members in C), each of which can be of any type. For each field/member, the record/structure definition specifies a field/member name identifier and the field/member type. For the remainder of this discussion, the Pascal terminology will be used.

VARIANT RECORDS/UNIONS. When the Pascal or C compiler allocates space for variant record fields, every field in the variant section is allocated the amount of space necessary to accommodate the largest variant field. Padding is used to fill up unused space for those variant fields which are smaller than the largest variant field.

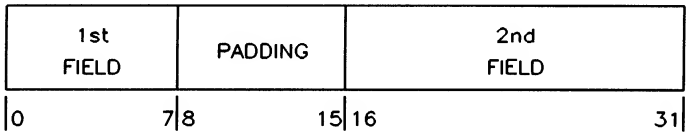
If a variant field is to be displayed, the specific name of the variant field should be specified in the measurement. If it is not (i.e. the record as a whole is specified), the analyzer will display the record in terms of the first (C source code) or last (Pascal source code) variant field. Therefore, padded space could be displayed as meaningful contents or vice versa.

Using the following program example, tracing AREC.SECOND would display a 32-bit value. However, if AREC was traced, the display would show the 16-bit field in AREC followed by 48 bits shown as padded space.

```
EXAMPLE:  (Pascal)
  TYPE
    VREC = RECORD
             CASE TAG : INTEGER OF
                 1:  (FIRST: A_16_BIT_SIZE);
                 2:  (SECOND: A_32_BIT_SIZE);
                 3:  (THIRD: A_64_BIT_SIZE);
           END;
  VAR
    AREC : VREC;
    VAR1 : A_16_BIT_SIZE;
    VAR2 : A_32_BIT_SIZE;
    VAR3 : A_64_BIT_SIZE;

  BEGIN
   :
   .
  CASE AREC.TAG OF
      1:  AREC.FIRST := VAR1;
      2:  AREC.SECOND := VAR2;
      3:  AREC.THIRD := VAR3;
  END;
   :
   .
  END;
```

During compilation, padding may take place automatically to handle memory alignment. For example, in a record defined as having an eight-bit field followed by a 16-bit field, the eight bits between the two fields may be padded to accommodate word boundaries.



If the entire record was displayed, it would contain an 8-bit field of padding. See chapter 8 for an example of how pad bytes are displayed.

Chapter 16

OPERATIONAL THEORY

OVERVIEW

This chapter provides information on the operational theory of the software analyzer to aid you in understanding why some measurements function as they do. This chapter provides the following information:

- A description of high level constructs that the analyzer uses.
- A discussion of software analyzer recognition resources/counters.
- Measurement operational theory.
- More on resource allocation.

GENERAL INFORMATION

This chapter describes the operation of the software analyzer. Performing high level software analysis is not a simple task; complex hardware and software are required, and tradeoffs must be made. Understanding how the software analyzer accomplishes measurements and uses its resources will enable you to use the software analyzer more effectively.

Appendix C contains information on stack architecture and memory structures that will help you understand how the software analyzer functions.

HIGH LEVEL LANGUAGE CONSTRUCTS

To understand how the software analyzer works, you need to understand the definitions of some high level language constructs with which the software analyzer interacts. The following paragraphs describe these high level constructs.

Procedures

Procedures have defined entry and exit points and can be associated with both their source code line numbers and their physical addresses at link time.

Variables

From the viewpoint of the software analyzer, variables are considerably more complex than procedures. Variables can be of different types and can have more that one data element, e.g.,

arrays and records. Variables can be pointers, which do contain not data but addresses where data (or other pointers) can be found.

All types of variables can also either be static or dynamic. Static means that their location is fixed, i.e., a static variable is associated with a fixed address (or addresses) by the linker. A dynamic variable, on the other hand, is allocated at run time, either using the stack or the heap (the heap is an area of memory allocated at link time just for this use). Variables defined local to a procedure are dynamic; they are given a location on the stack when the program is executed.

Tracing dynamic variables is more complex than tracing static ones, especially in real-time. The software analyzer cannot be set up to look at a dynamic variable until the variable is defined (scoped). This doesn't occur until run time. Thus, the software analyzer has specialized hardware which can be initialized "on the fly". These dynamic recognition resources are separate from the static resources in the software analyzer; these hardware resources cannot be shared.

The software analyzer can trace all types of variables. The data pointed to by a pointer, however, can only be accessed in the non-real-time trace data flow and display variable measurements. Seven levels of indirection are supported in these two measurements, i.e., the analyzer can display the data value being pointed to by a string of six pointers. The other measurements can only display the address located in the pointer location.

Symbols

A symbol is a procedure name, a program line number, or a variable name. An array or record is considered one symbol if traced as a whole, but if individual elements are traced, each counts as one symbol. For each measurement, up to ten symbols can be entered when the setup is being specified.

RECOGNITION RESOURCES AND COUNTERS

The software analyzer has 18 static low level recognition resources and four dynamic low level recognition resources. A low level resource is defined as an "equate". In other words, an IC chip is used to watch the emulation bus and compare address/status or data bit patterns with a predifined pattern. It is important to understand these low level resources and how each measurement utilizes them, as sometimes they can be used up.

The counting and timing is done with a 20-bit floating point grey code counter. These 20 bits make up the time tag for every stored state. The counter has a 100 nS accuracy.

TRACE MEASUREMENT THEORY

Each software analyzer measurement uses the low level recognition resources to look for specific data, addresses, or ranges. Whenever possible, the number of resources used is minimized; adjacent variables and modules can sometimes use just one range resource. Adjacent variables or modules must be adjacent in the emulation/user memory, not necessarily adjacent in code.

Trace Modules Measurement

Figure 16-1 shows the resources used in a "trace modules PROC1 , all file FILE_A , PROC4" measurement. Both address ranges and data equates are used to detect module entry and module exit. The data equates are what actually determine these entry and exit points. Using "hooks" provided by the compiler, these equates are set up to recognize the first and last instructions of a module. The way HP compilers are written, unique instructions indicate entry or exit points of modules. Address ranges are then used to qualify the entry and exit points so that only the ones in the specified modules are saved.

Address ranges are set up around the addresses associated with the specified modules; they cannot include code space for another module or else the trace would catch unwanted data. In this example, even though three symbols were specified, only two ranges are needed. All modules in a single file are adjacent, and it happens that PROC4 is adjacent to FILE_A.

Finally the results of the recognition resources are ANDed together, data associated with an entry or exit point is stored in trace memory. With processors that use prefetching, sometimes data is stored which is not a true entry; The software analyzer detects this when it postprocesses the data and filters out these points.

Ten explicitly named modules can be traced. This limitation is imposed by the overall 10 symbol limit in trace specifications. However, if the specified modules are not adjacent, only four can be traced (there are only four range resources). If all the modules in a file are specified, up to 255 different modules can be traced.

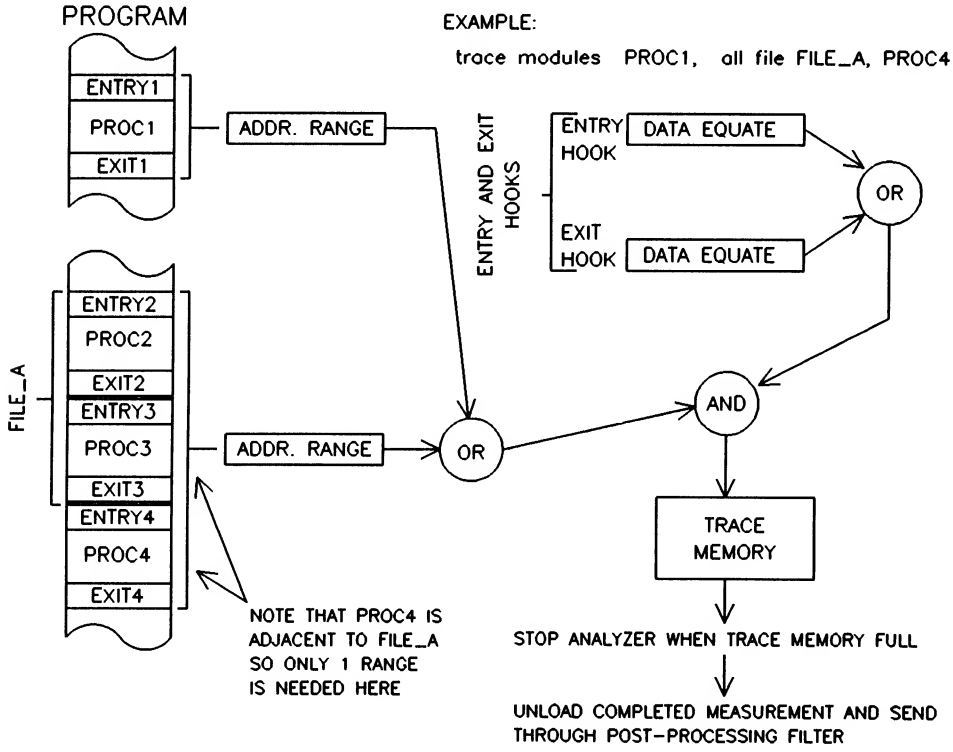


Figure 16-1. Trace Modules Measurement Diagram

Trace Data Flow Measurement

Figure 16-2 shows the resources used in the trace data flow measurement. This measurement can only run in non-real-time mode because the software analyzer needs information that is not available on the emulation bus. Address equates are used for each module, limiting the number of modules that can be specified to three. For each module, one equate is used for the entry and one for the exit. Another equate is used for each module, to recognize an address at the end of user code, just before the address of the module exit. This accommodates tracing infinite recursion.

Upon entry to a module, a frame is created. This frame contains information required by the analyzer, such as where the stack is located. Up to 256 frames can be stacked at a time. Whenever a recursive routine calls itself, a new frame is created and stored. When the recursive routine returns, the old frame is available at module exit.

This 256 state memory limits some measurements to only only 255 levels of recursion. In trace data flow measurements, since the emulator can be halted, frames can be created anywhere within the module. By having an equate set up to look for an address that occurs within a module, but before the exit, a frame can be created when this address is recognized. This address is chosen to be at the end of the user code so that any calls would have occurred previous to the address. Therefore, unlimited recursion be supported. Anytime frame information is lost because the level of recursion exceeds 255, it can be recreated when returning to the module.

For this reason, three equates are needed for each module being traced. When any of these equates are satisfied, the analyzer is halted, and the values of specified variables are read from emulation or user memory, and stored in trace memory. Then the analyzer and emulator is restarted.

The only limit imposed on the number of variables specified, is the 10 symbol limit on entering the measurement. Variables that are not scoped at module entry or exit cannot be traced.

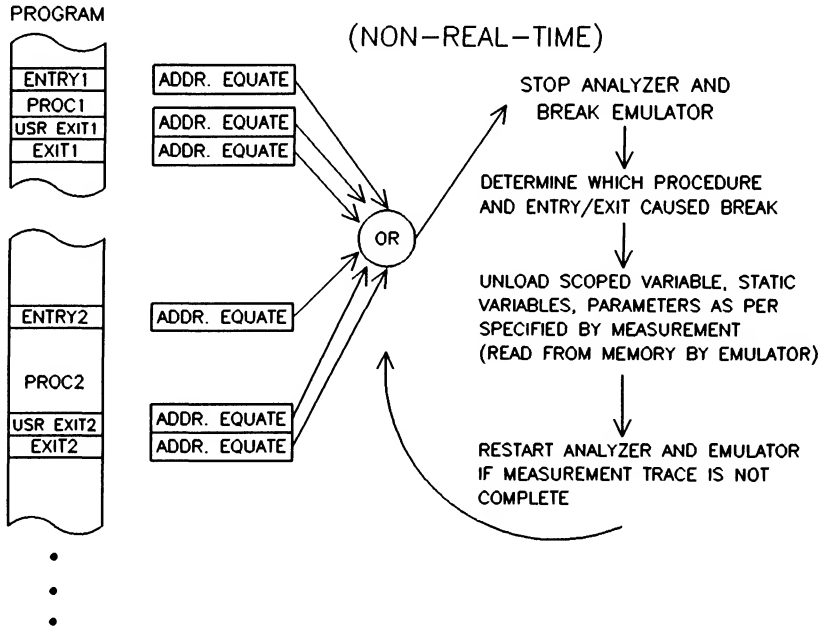


Figure 16-2. Trace Data Flow Measurement Diagram

Trace Variables Measurement

The trace variables measurement functions the same way in real-time as in non-real time. Figure 16-3 shows the resources used in this measurement. Static Variables are comparatively easy to trace. Address equates are used for any static variable that is one byte wide, and ranges are used on longer variables such as records and arrays. The analyzer will also use one range over any adjacent variables. If the range is accessed, the address and values are stored.

Dynamic variables function in much the same way. However, since the number of dynamic recognition resources is less, one range is used to cover all variables, even if they are not adjacent. During postprocessing, the unwanted accesses are filtered out. The software analyzer must locate the stack reference and the actual memory locations for these variables, which is not defined until the program is executing.

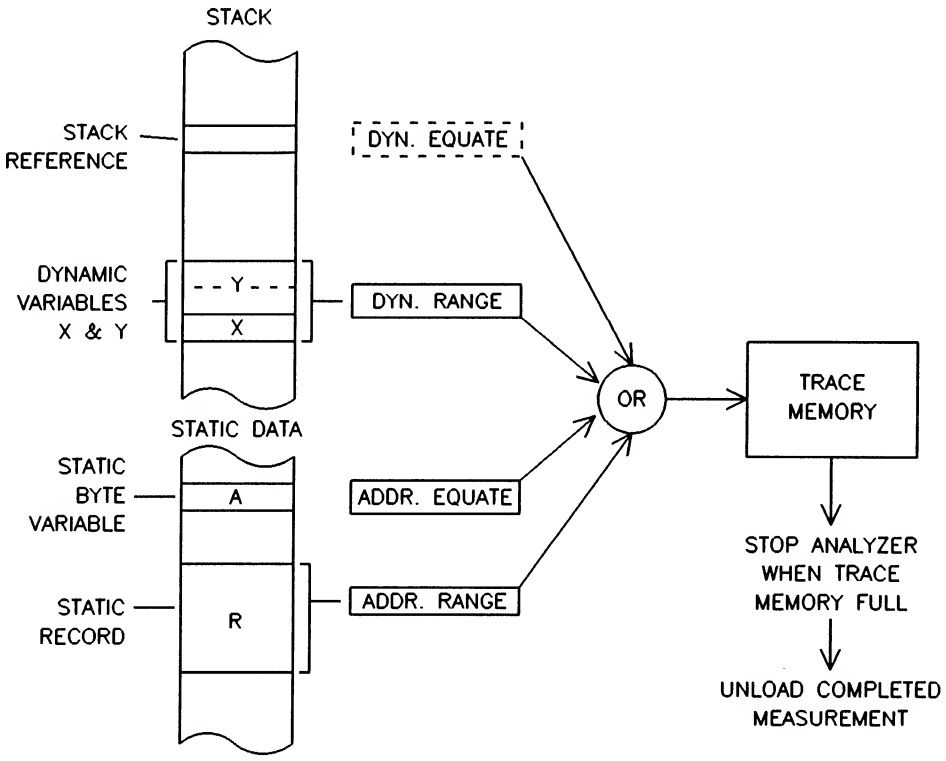


Figure 16-3. Trace Variables Measurement Diagram (Dynamic and Static)

Figure 16-4 shows how the dynamic variable locations are determined by the software analyzer. One dynamic equate is used to locate the entry to the procedure where the dynamic variables of interest are defined. From this point the software analyzer can determine the first data write onto the stack. This provides an immediate stack reference, and is stored in trace memory. From the database file, the offsets of the dynamic variables are known. Therefore, locations of the variables can be loaded using this stack reference and and offsets to the dynamic range resource. This is done before any of the variables have been read or written to by the user program. The dynamic equate is then reloaded, enabling the software analyzer to recognize the location on the stack of the procedure's return address. All accesses to the range of locations are stored in trace memory until the equate indicates that the return address has been read (popped off the stack). After this point the variable no longer exists and the sequence is started up again.

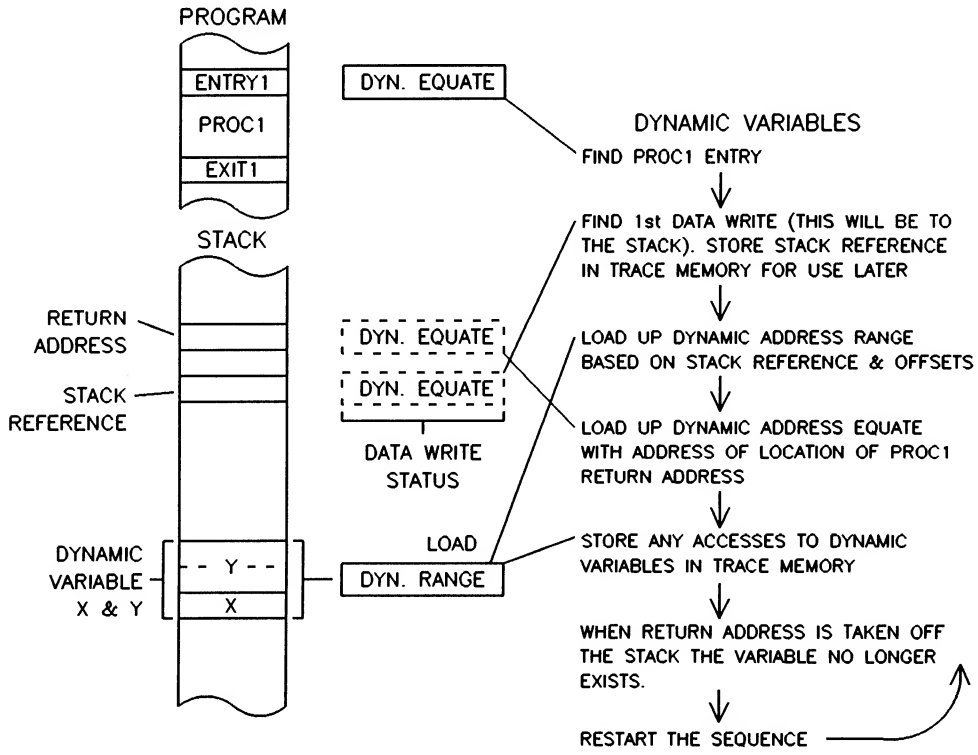


Figure 16-4. Trace Variable Measurement Diagram (Non-Real-Time and Real-Time)

Trace Statements Measurement

Trace statements can be viewed as two measurements , a real-time measurement and a non-real-time measurement. The real-time measurement is less complex conceptually because the emulator is never broken, and all information is flowing over the emulation bus. Figure 16-5 shows a real-time trace statements measurement over procedure PROC1. One address range resource is used to detect the window start and stop points. The special address equate which is set up to detect an opcode is used to detect the execution of code. These two resources are ANDed together. When the two resources are true, a "window" is is opened and all the data flowing over the emulation bus is stored in trace memory. When the address range signals that the program is out of the specified line range, the storing "window" is closed. A trace statements using the "don't care" specification simply causes the analyzer to execute with the window continuously open. During postprocessing, the data is interpreted and the lines are displayed in their executed order.

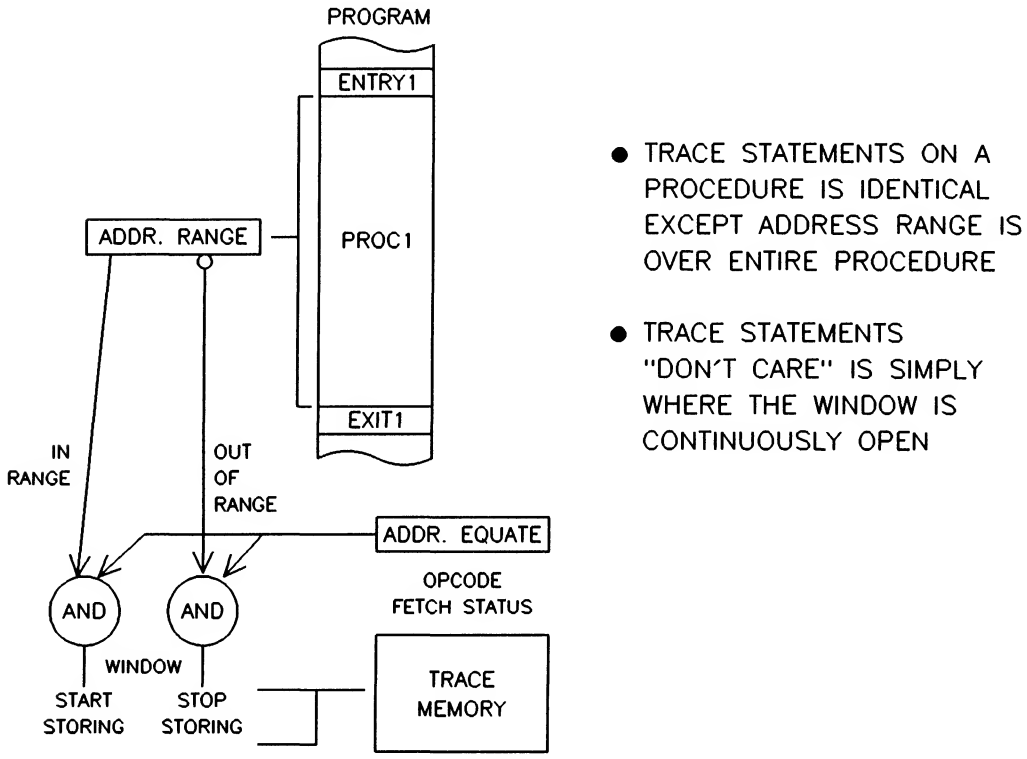


Figure 16-5. Trace Statements Measurement Diagram (Real-Time)

In non-real-time (figure 16-6), the trace statements measurement provides more data to the user. This is a more complex measurement. Because the analyzer can halt the emulator and determine where the stack is (create frames), dynamic and local variable values can be captured. The use of the one address range to determine a storage "window" is the same in non-real-time, but in addition, two address equates are used to detect entry and exit points. Whenever the program enters the specified module, the emulator is halted and the new stack information is stored for postprocessing later.

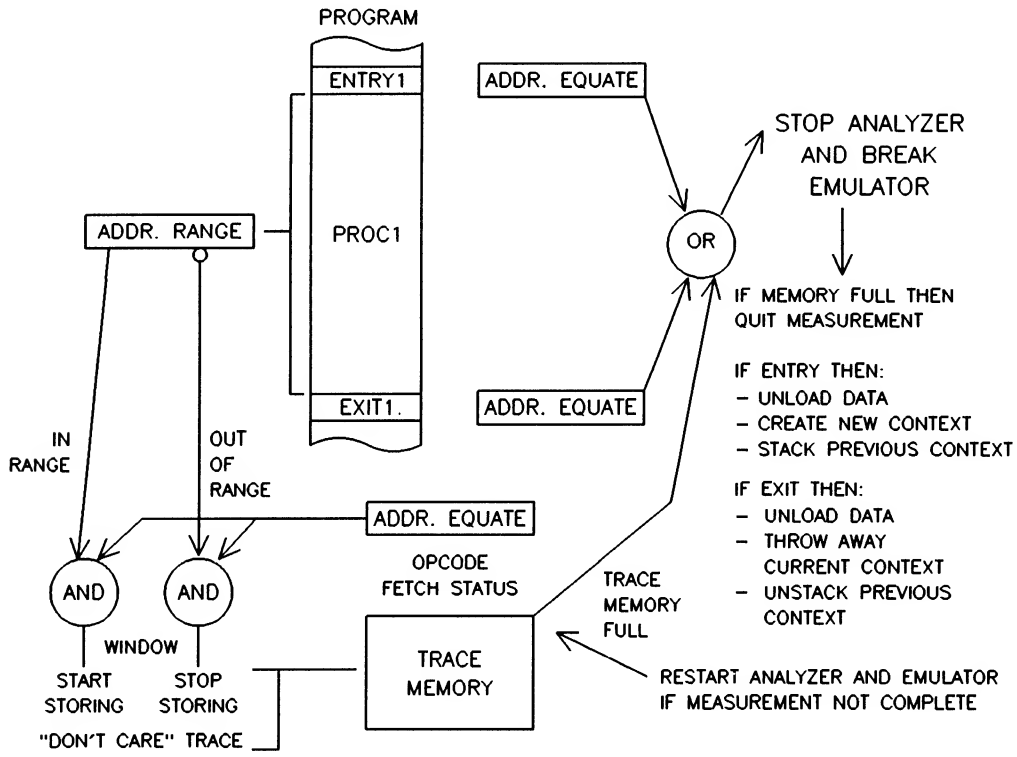


Figure 16-6. Trace Statements Measurement Diagram (Non-Real-Time)

Count Statements

The count statements measurement uses the same address range resource as the other measurements, but also has some dedicated hardware. The measurement can trace 255 lines in one module, but the traced portion of the program cannot exceed 4K bytes of memory. The reason is that one of the dedicated functions is a 4K to 256 "bucket" mapper. The "bucket" refers to the 12-bit counters associated with each source line. The measurement works by assigning the address range resource over the specified module/line range. Before the measurement is executed, the mapper is loaded, using the line number information found in the comp_db file. When the measurement is executed, the appropriate counter or "bucket" is incremented when the first machine code statement of a given line number is executed. Thus a line that contains many instructions (which could even loop and execute a number of times) is incremented only once.

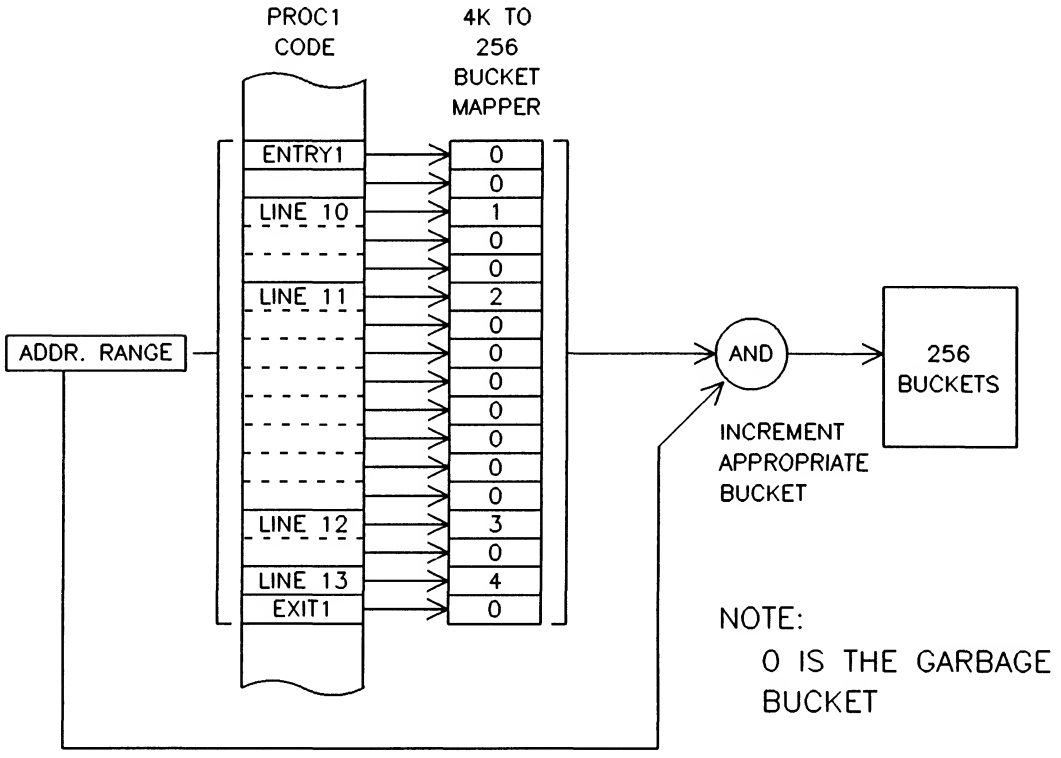


Figure 16-7. Count Modules Measurement Diagram

Time Modules

The time modules measurement uses two address equates for each module. Thus, a limit of four modules can be traced. These equates simply look for entry and exit points, and when true store the state. The 20-bit counter is started at the beginning of the measurement, and the absolute time for every exit and entry point is saved as a time tag with the stored state. When the measurement is complete, these time tags are used to determine time spent in each module, and then the statistical results are determined.

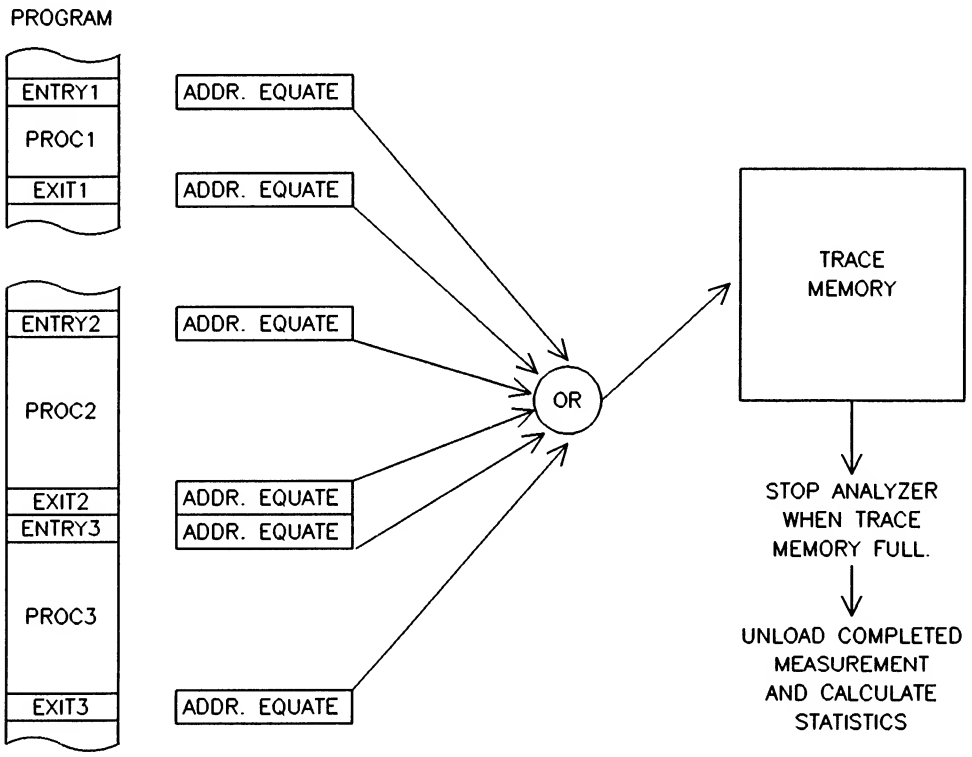


Figure 16-8. Time Modules Measurement Diagram

MORE ON RESOURCE ALLOCATION

Because of resource allocation, each measurement has different limits. A measurement's limits is also affected by the measurement enable and disable functions, which use the same range and equate recognition resources. This will cause measurement limits to vary from measurement to measurement. For each level of sequencing, one equate is used in either an enable or disable term. When in non-real-time, the enable equates can be reloaded before the measurement is initiated. Thus they do not take any resources away from the measurement specification. However, when specifying complex measurements in real time, the allocation of resources may involve tradeoffs.

Appendix A

OPERATING SYNTAX DIAGRAMS

INTRODUCTION

This appendix contains the operating syntax diagrams for the software analyzer. These diagrams are based on the guided-syntax softkeys that appear when the software analyzer is being used. The following syntax diagrams are provided in this appendix.

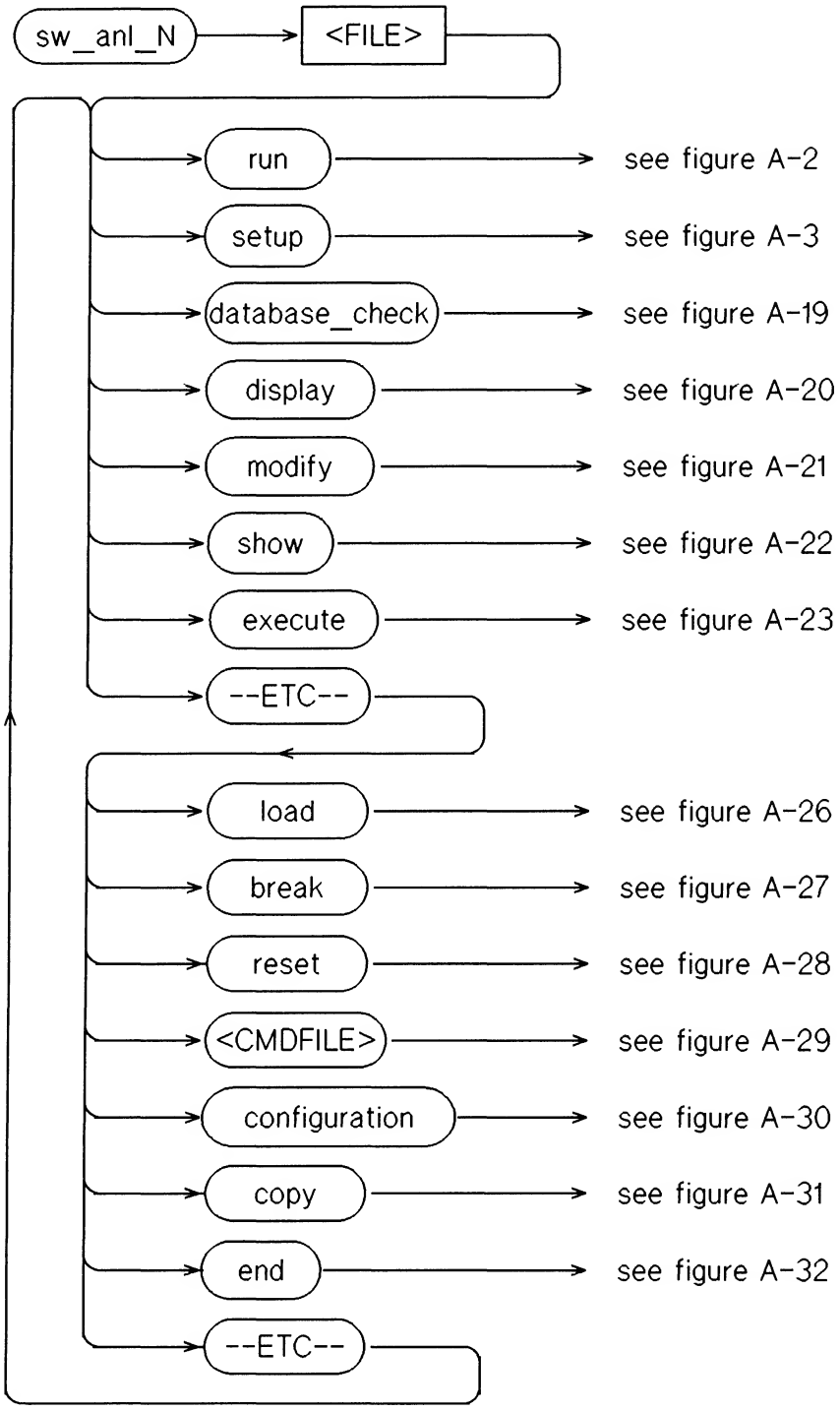


Figure A-1. Software Analyzer Level Syntax Diagram

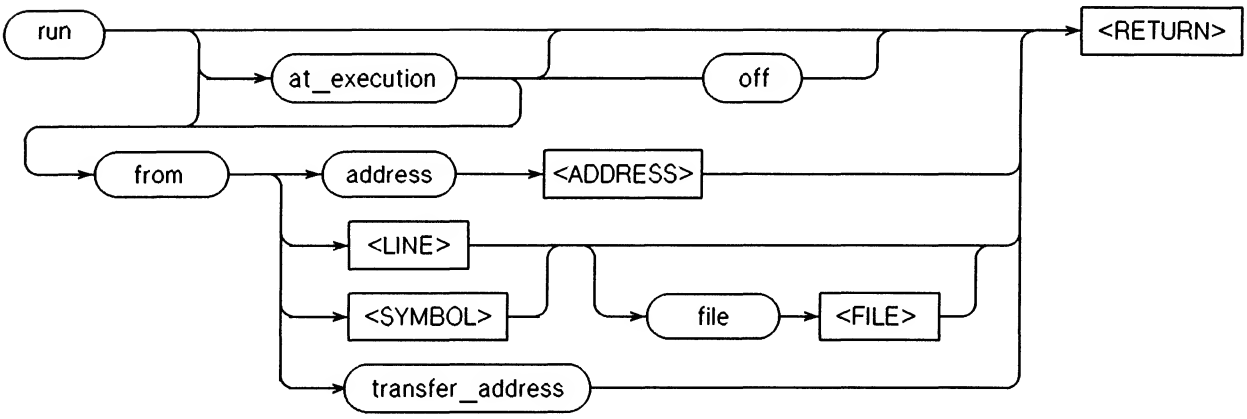


Figure A-2 Run Syntax Diagram

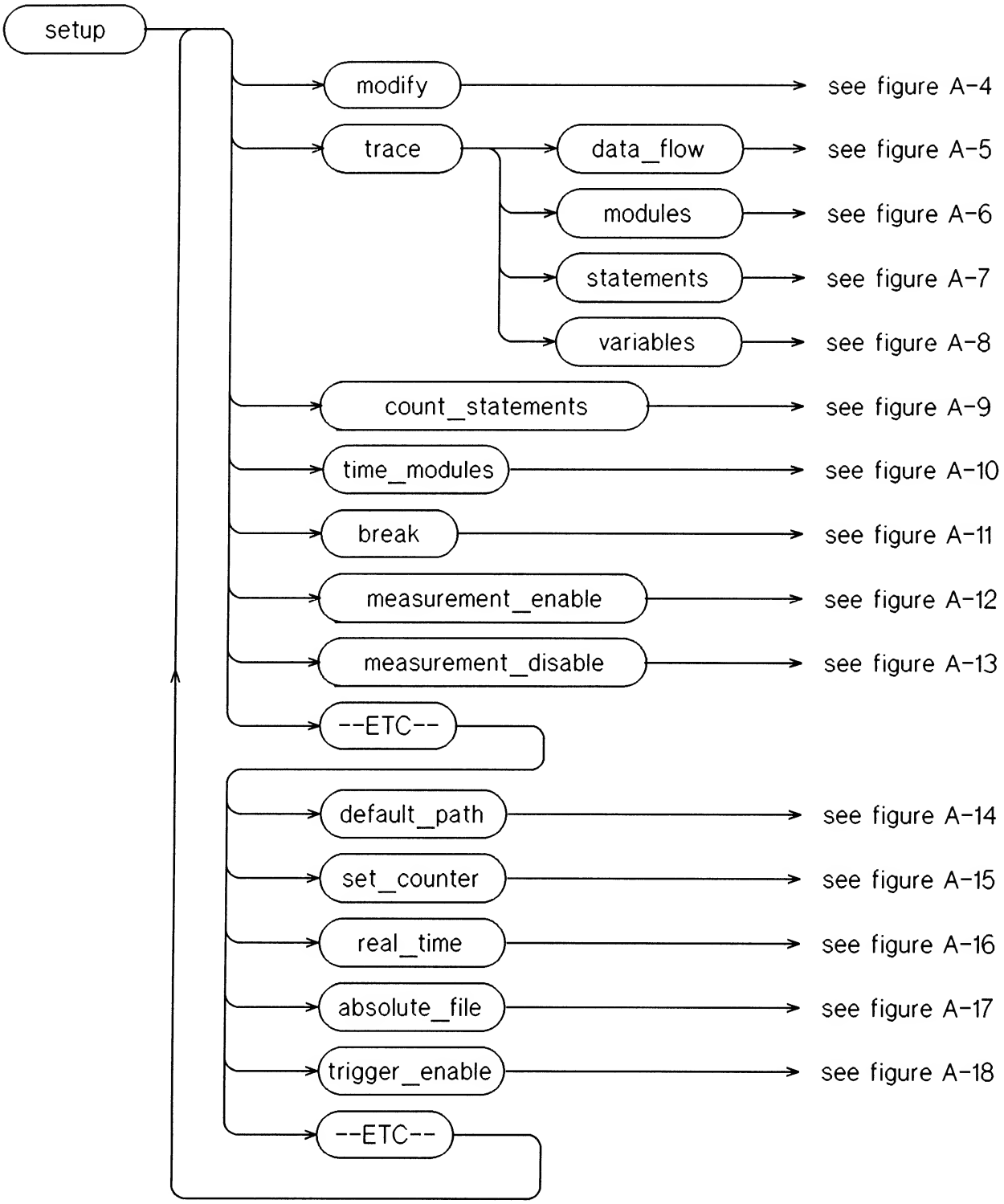


Figure A-3. Setup Syntax Diagram

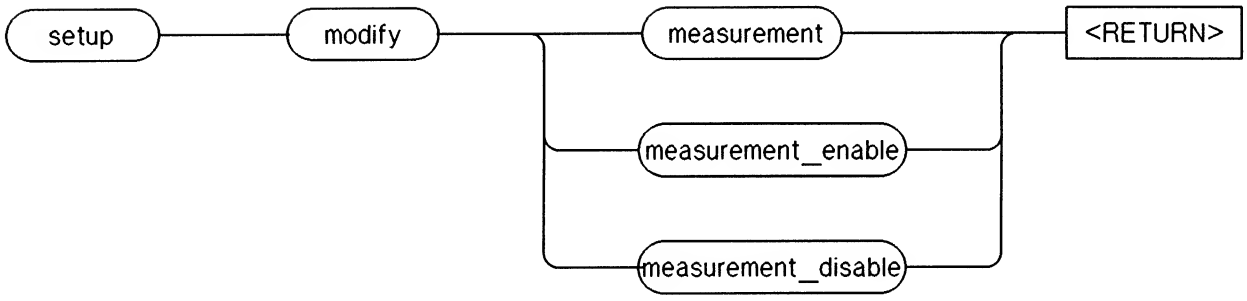
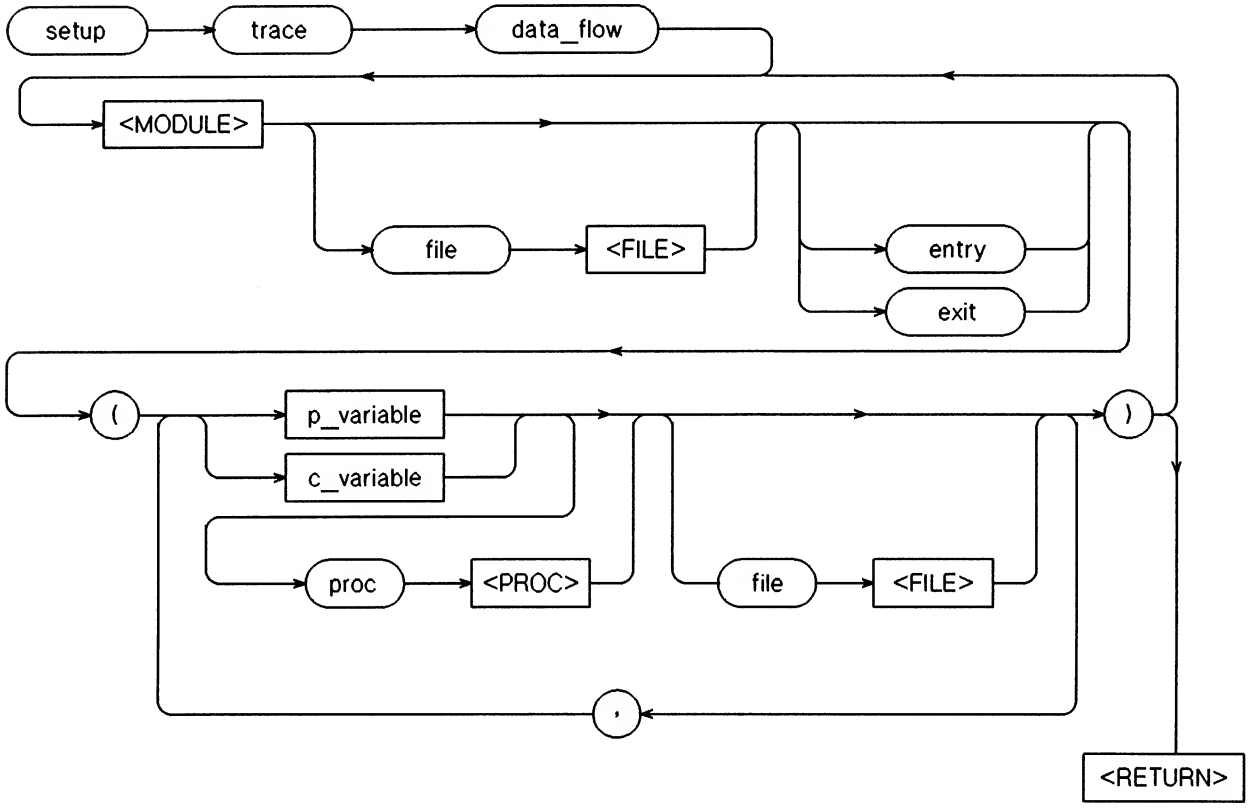


Figure A-4. Setup Modify Syntax Diagram



*See figures A-34 and A-35 for c_variable and p_variable syntax.

Figure A-5. Setup Trace Data_Flow Syntax Diagram

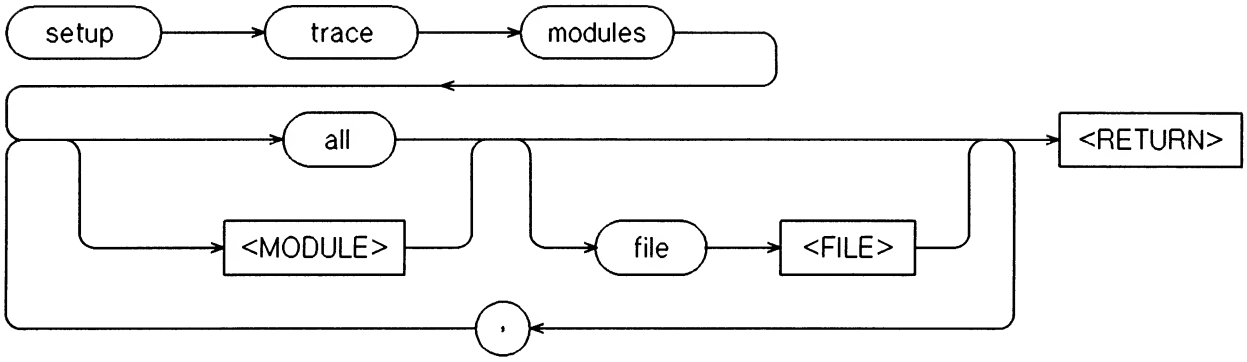


Figure A-6. Setup Trace Modules Syntax Diagram

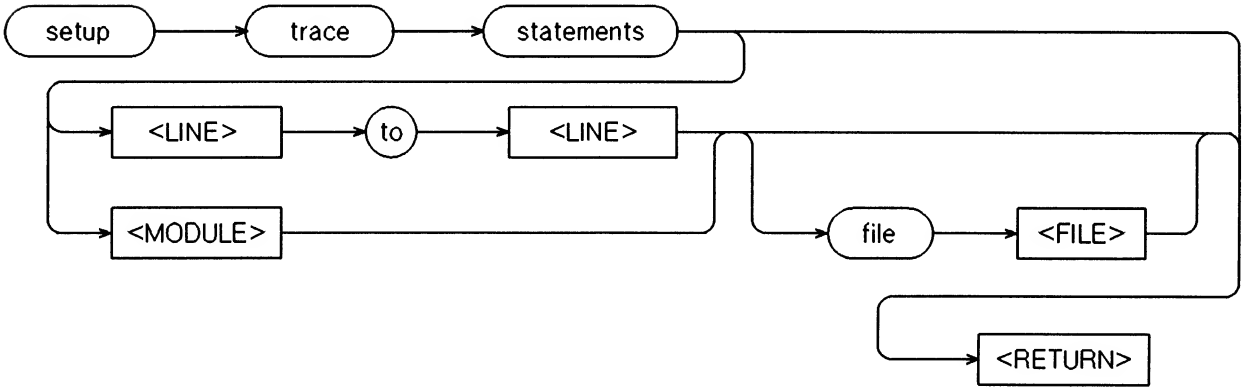
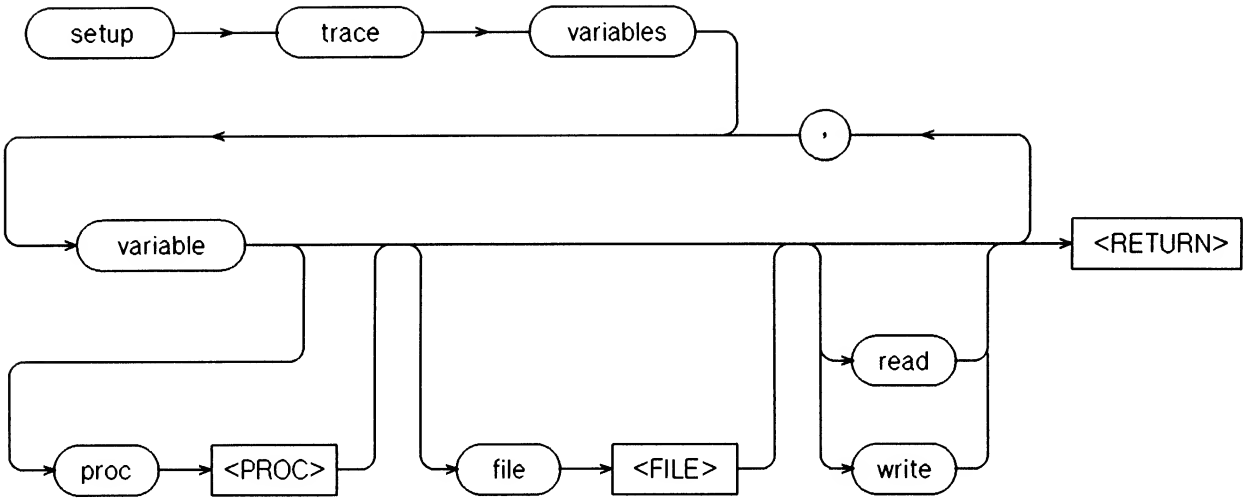


Figure A-7. Setup Trace Statements Syntax Diagram



*See figure A-33 for variable syntax.

Figure A-8. Setup Trace Variables Syntax Diagram

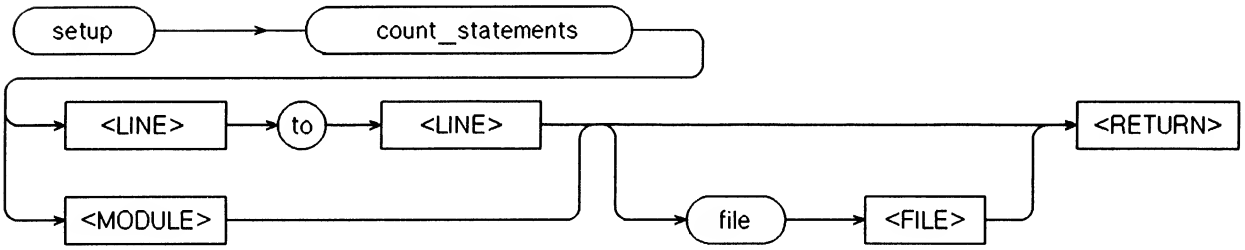


Figure A-9. Setup Count Statements Syntax Diagram

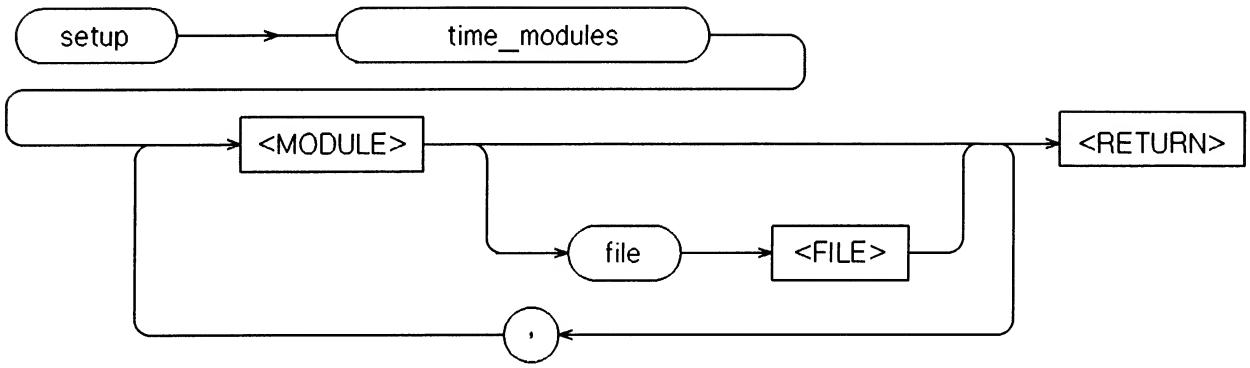


Figure A-10. Setup Time Modules Syntax Diagram

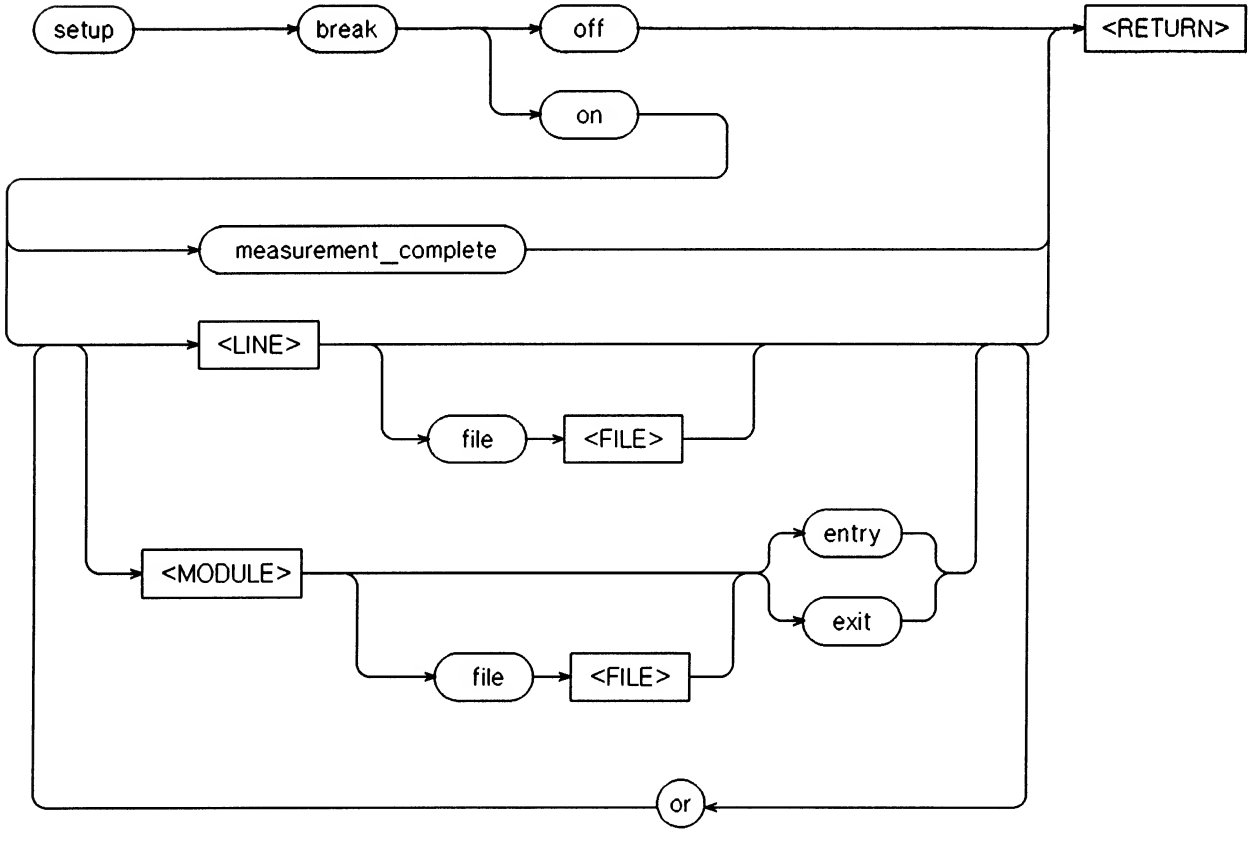


Figure A-11. Setup Break Syntax Diagram

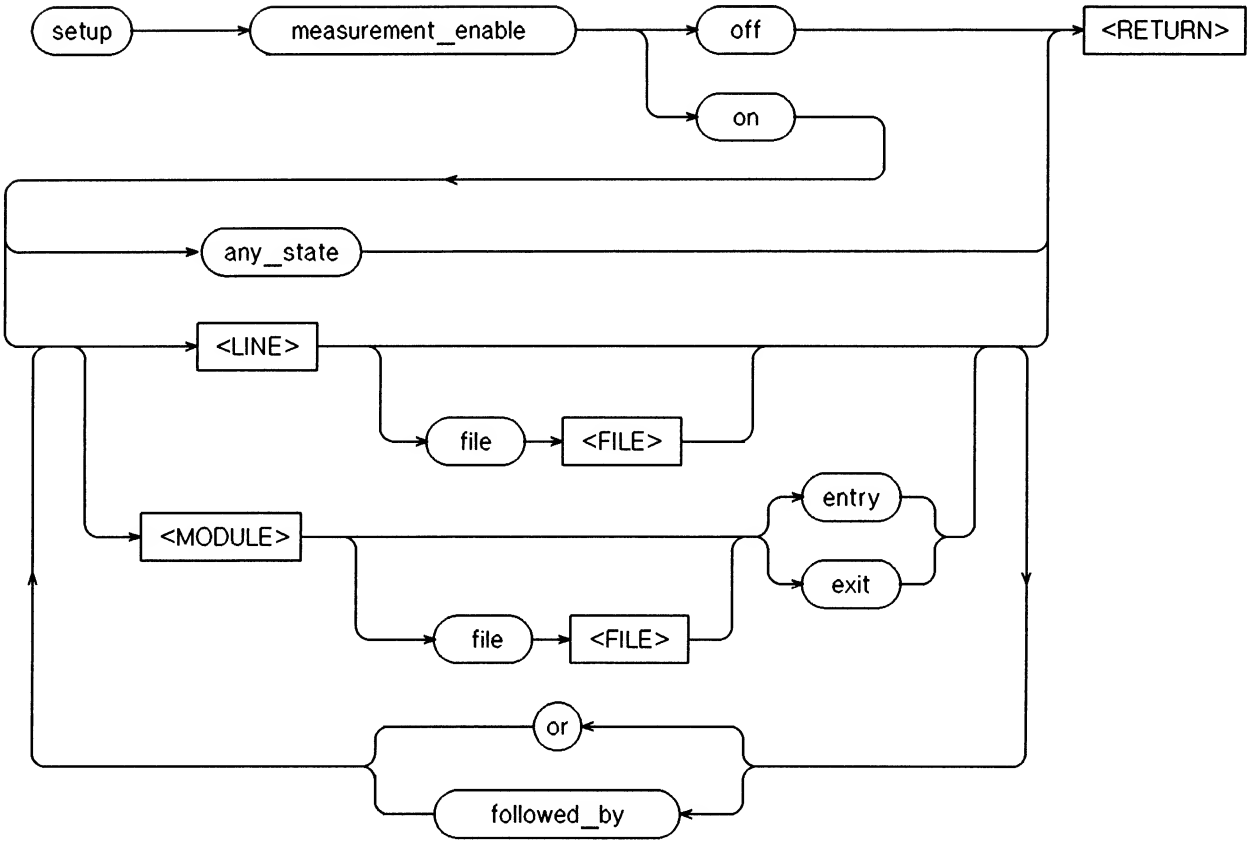


Figure A-12. Setup Measurement_Enable Syntax Diagram

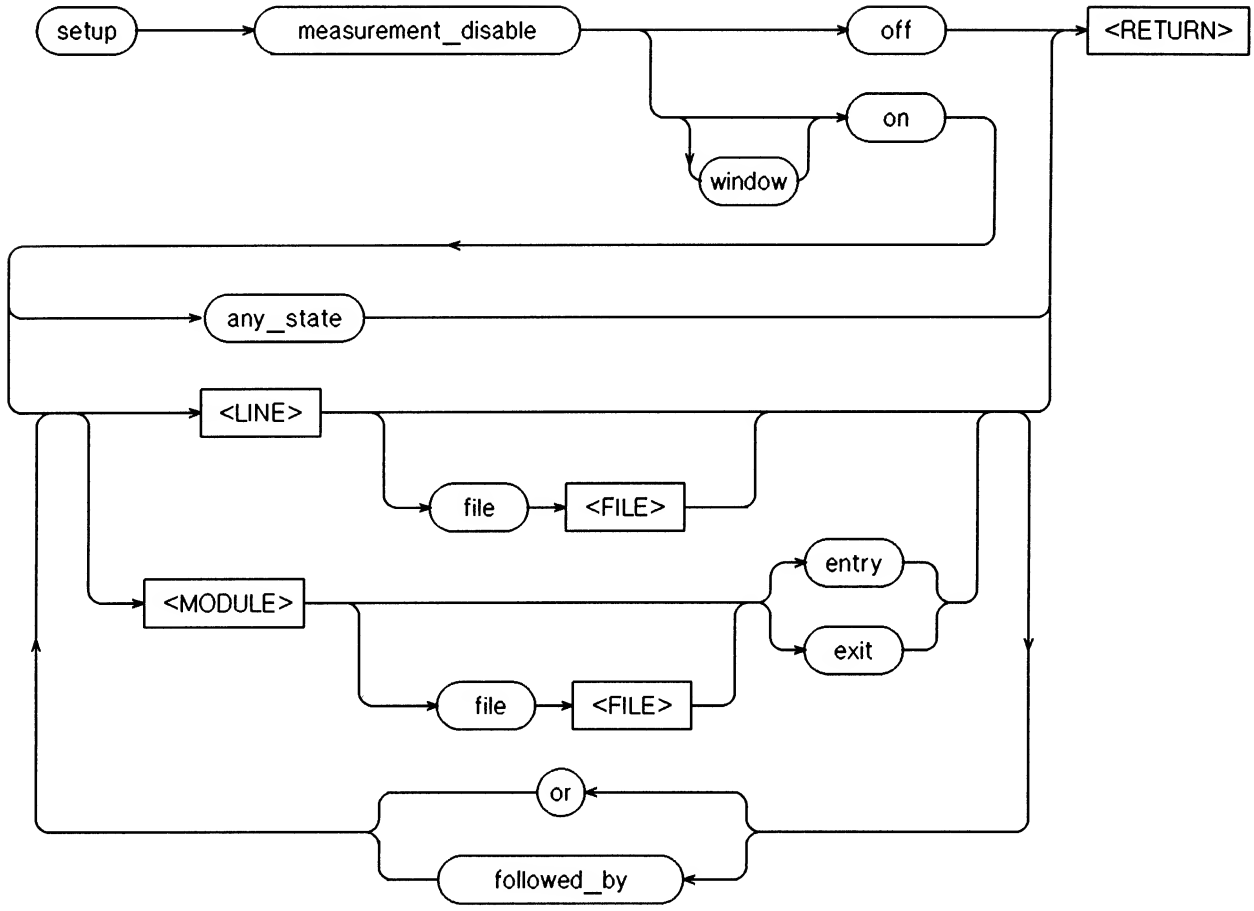


Figure A-13. Setup Measurement_Disable Syntax Diagram

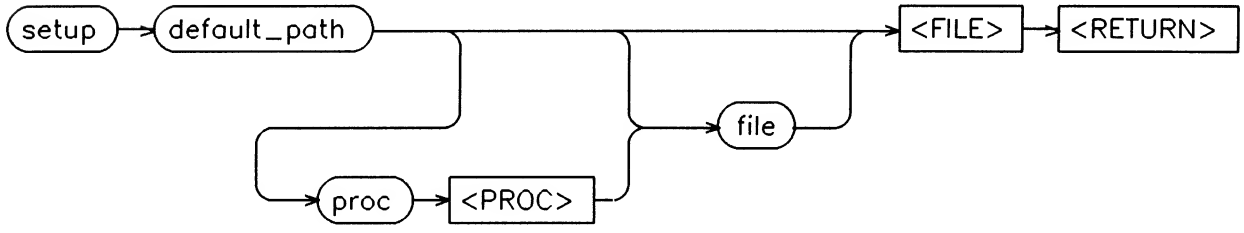


Figure A-14. Setup Default_Path Syntax Diagram

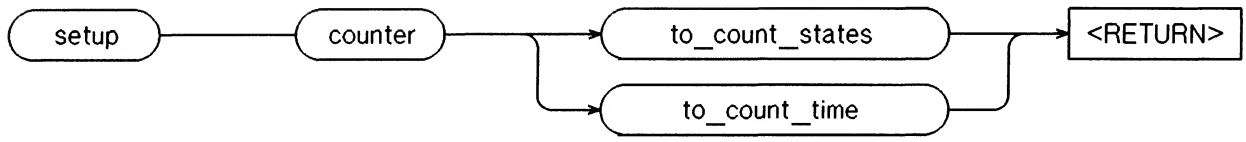


Figure A-15. Setup Counter Syntax Diagram

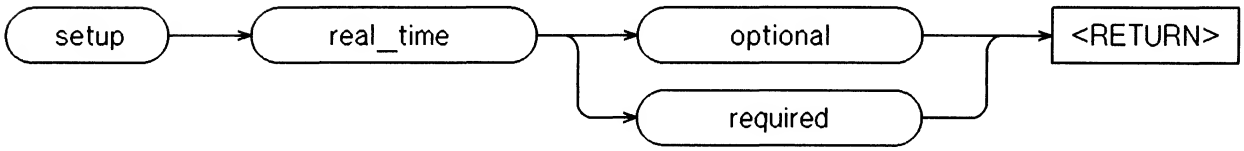


Figure 16. Setup Real_Time Syntax Diagram

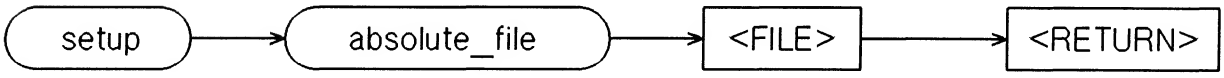


Figure A-17. Setup Absolute_File Syntax Diagram

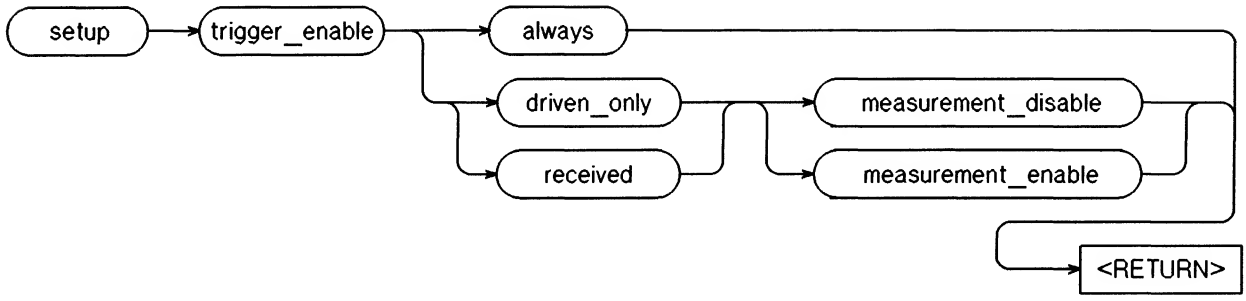


Figure A-18. Setup Trigger_Enable Syntax Diagram

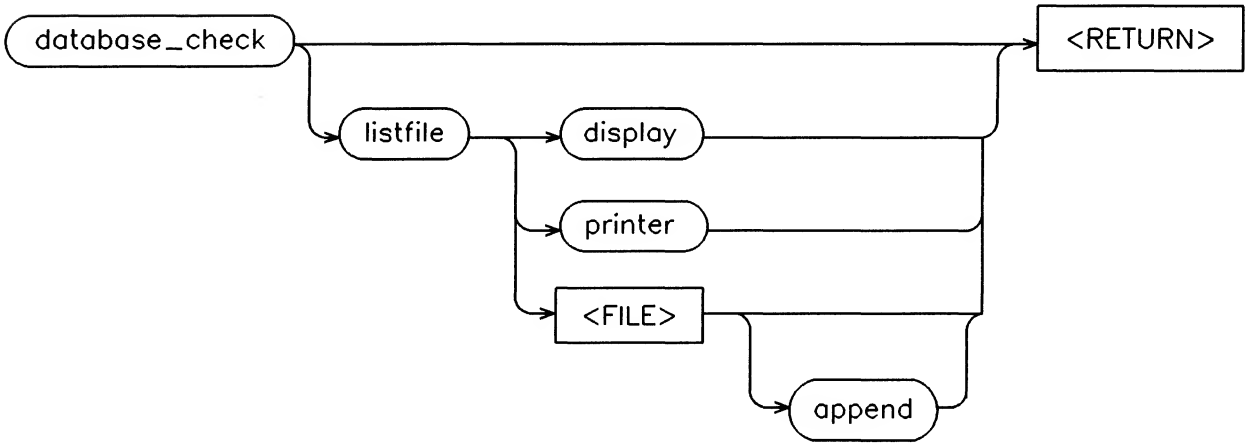
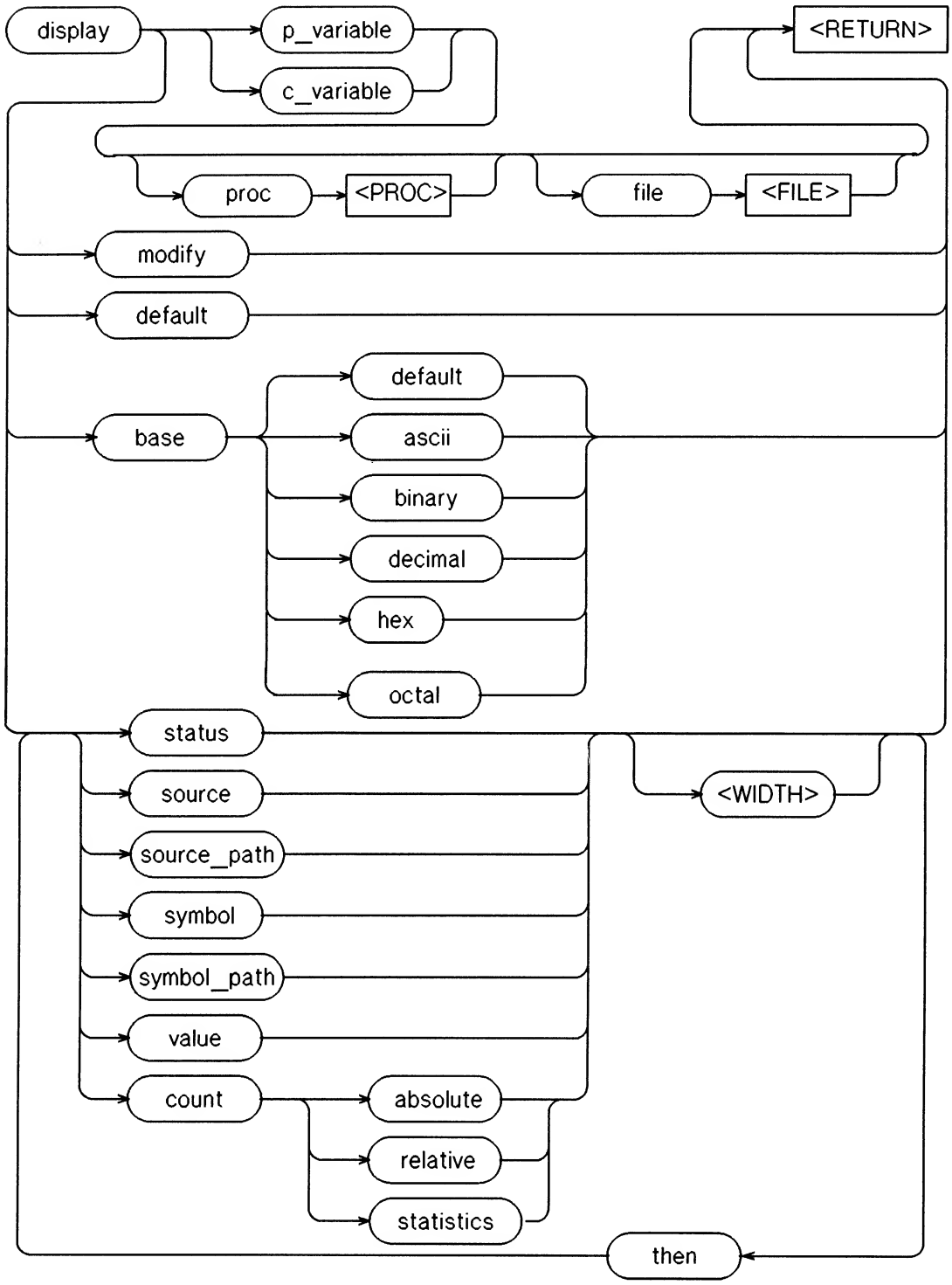
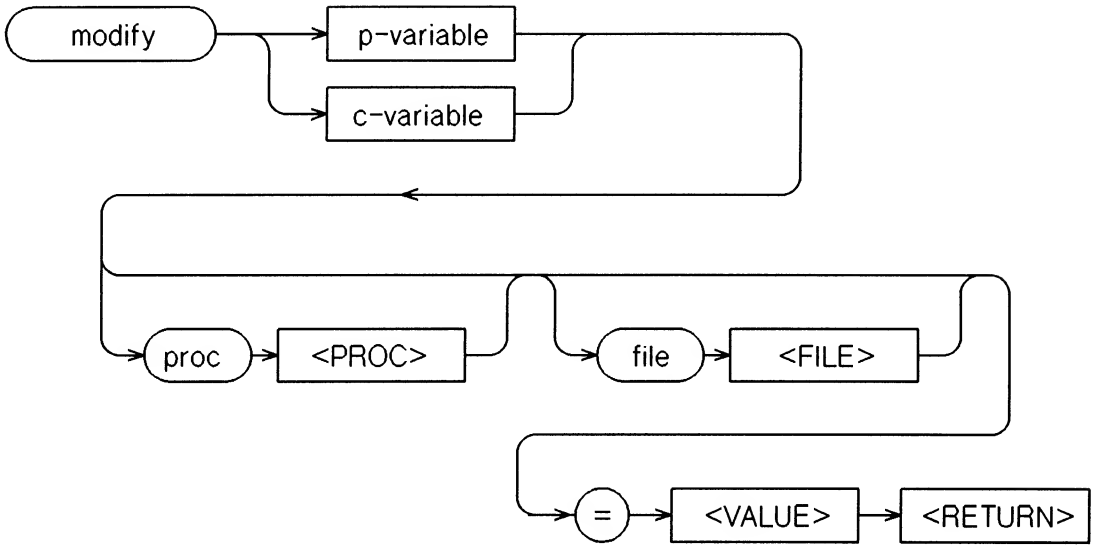


Figure A-19. Database_check Syntax Diagram



*See figures A-34 and A-35 for c_variable and p_variable syntax.

Figure A-20. Display Syntax Diagram



*See figures A-34 and A-35 for c_variable and p_variable syntax.

Figure A-21. Modify Variables Syntax Diagram

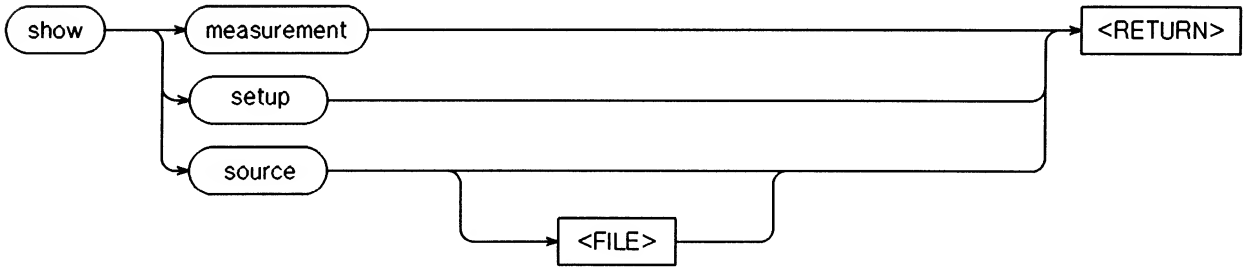


Figure A-22. Show Syntax Diagram

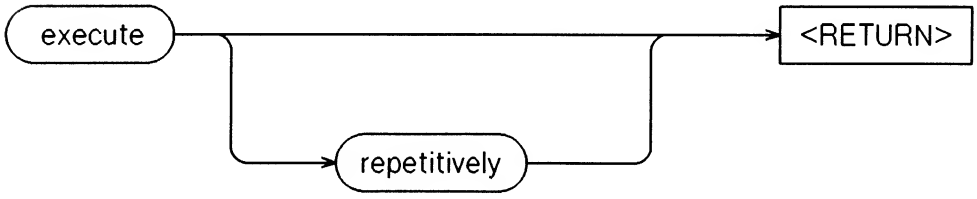


Figure A-23. Execute Syntax Diagram

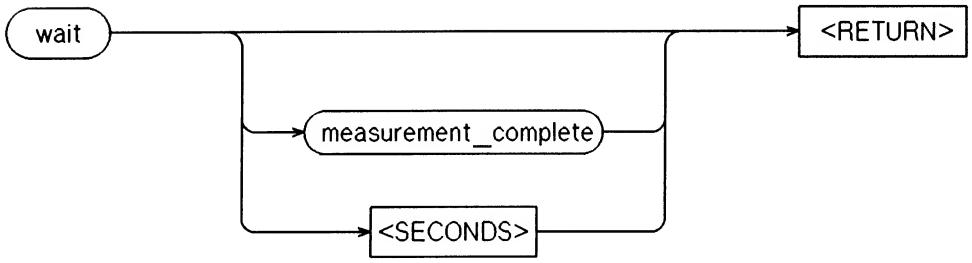


Figure A-24. Wait Syntax Diagram

Figure A-25. Halt Syntax Diagram

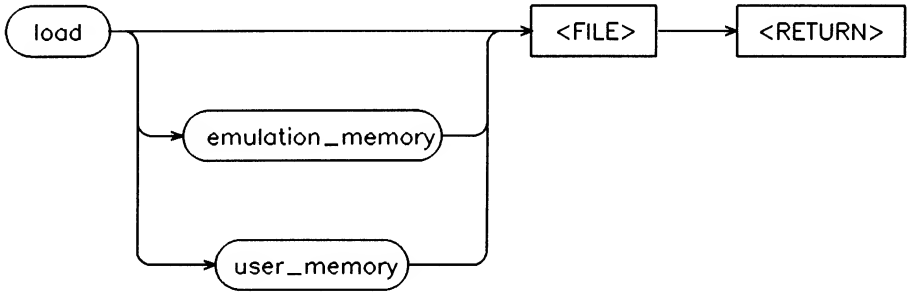


Figure A-26. Load Syntax Diagram

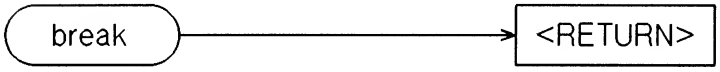


Figure A-27. Break Syntax Diagram

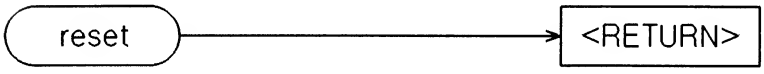


Figure A-28. Reset Syntax Diagram

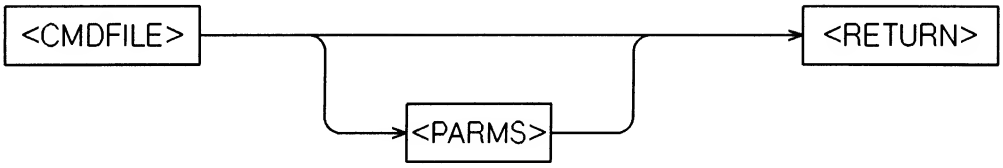


Figure A-29. <CMDFILE> Syntax Diagram

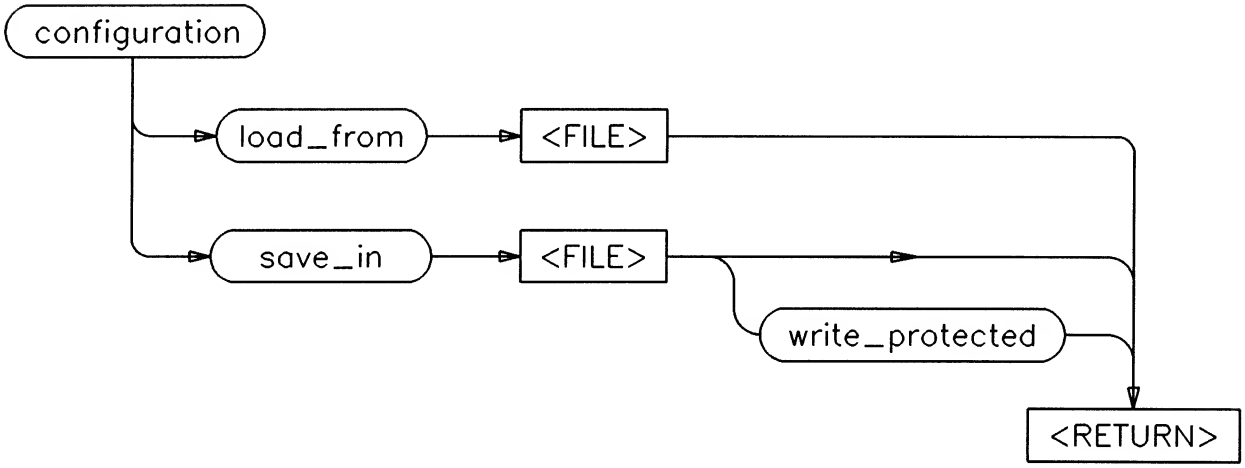


Figure A-30. Configuration Syntax Diagram

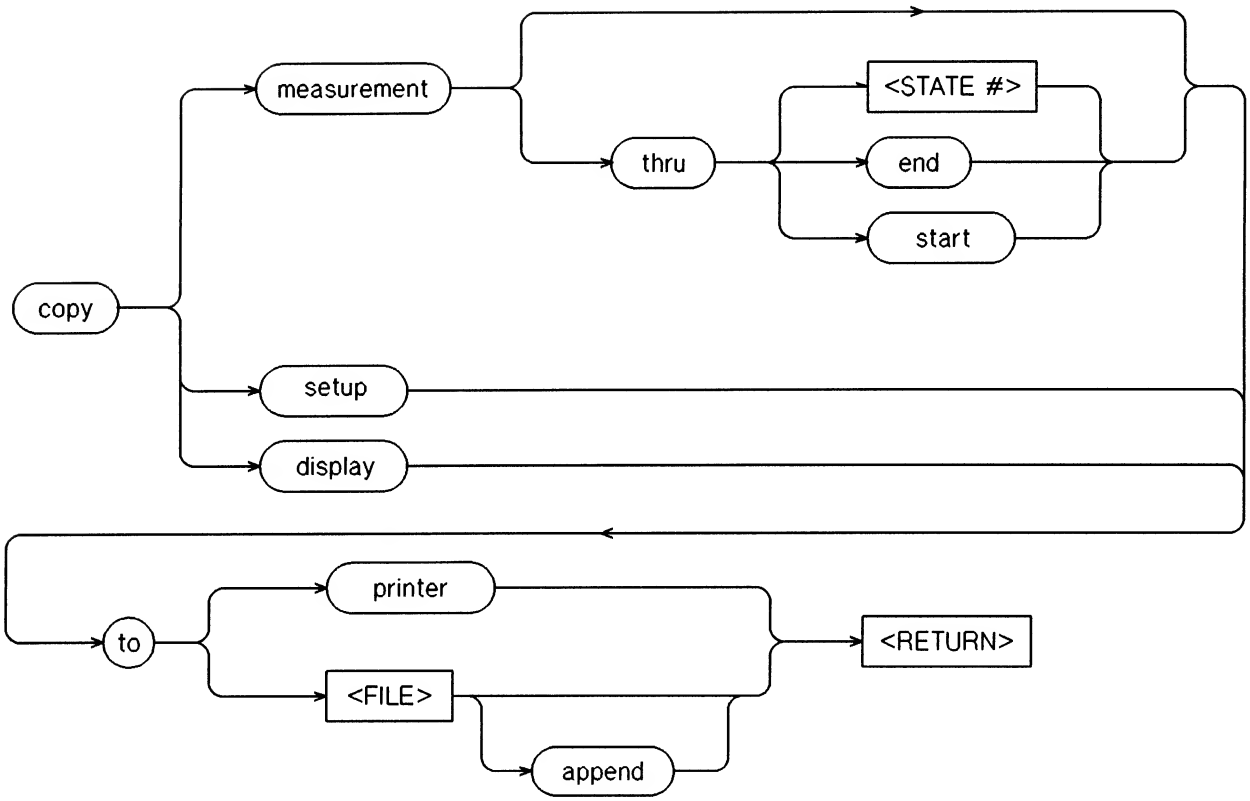


Figure A-31. Copy Syntax Diagram

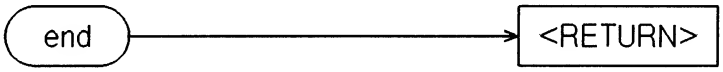


Figure A-32. End Syntax Diagram

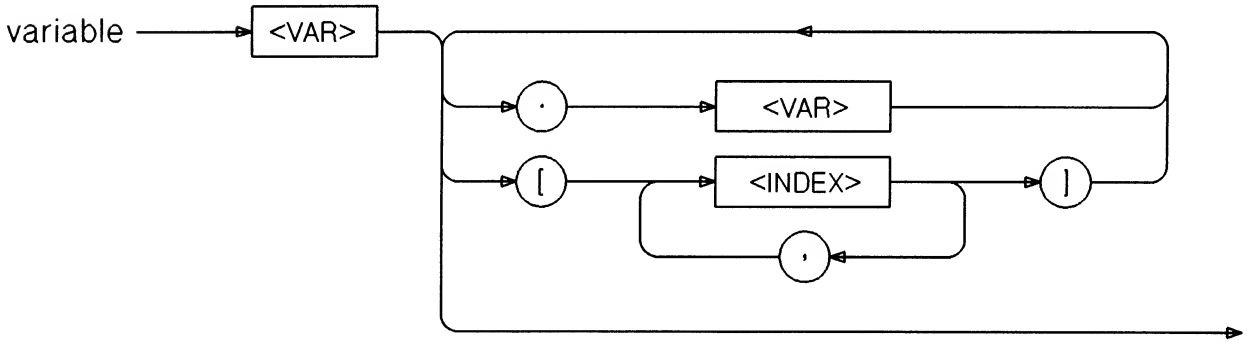


Figure A-33. Variable Syntax Diagram

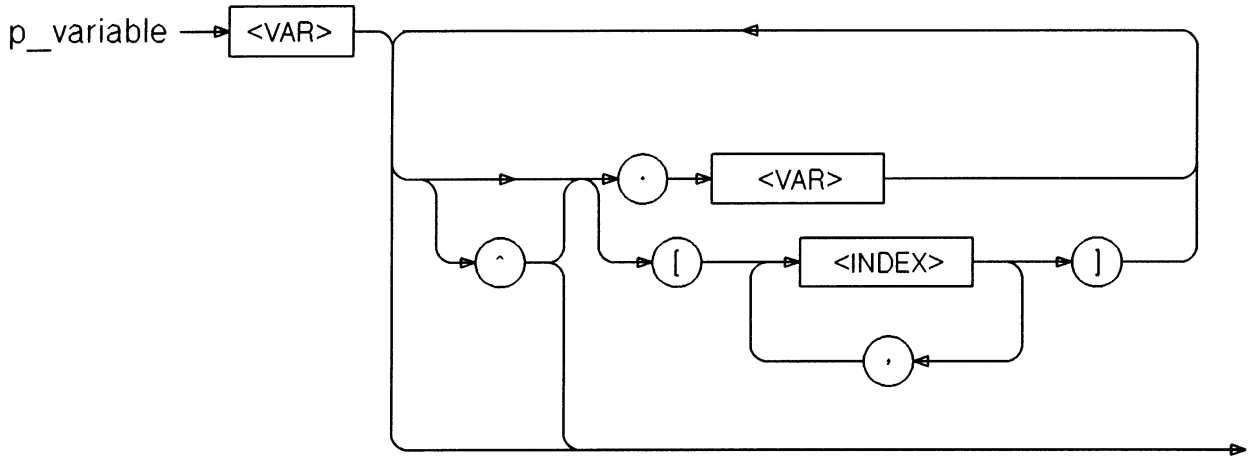


Figure A-34. Pascal Variable Syntax Diagram

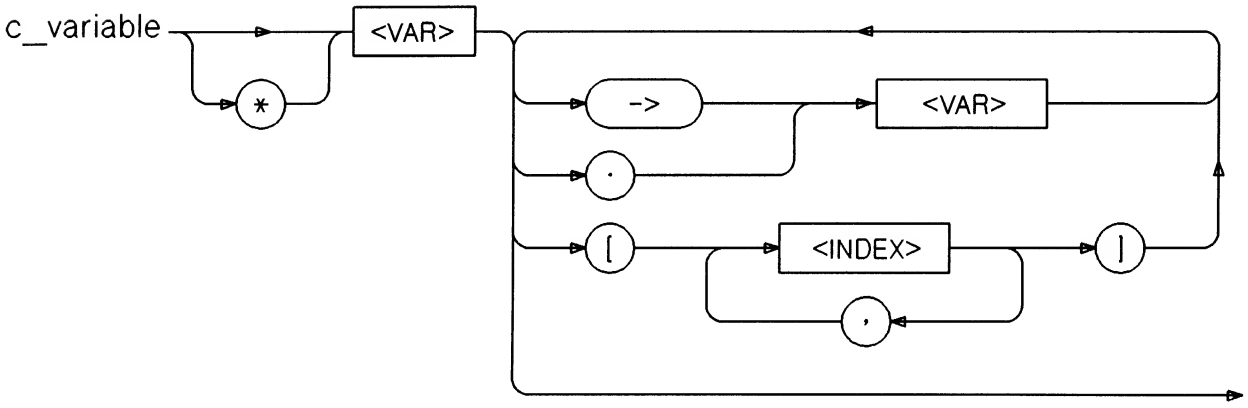


Figure A-35. C Variable Syntax Diagram

Appendix B

STATUS, ERROR AND SOFTKEY PROMPT MESSAGES

INTRODUCTION

This appendix contains a list of the status and error messages, and the softkey prompts and their corresponding messages. All these messages are displayed on the CRT as a result of the software analyzer software. An explanation of each message is given. Table B-1 provides a list of status messages, table B-2 provides a list of error messages, and table B-3 provides a list of the softkey prompts. Status messages are displayed on the screen to provide an indication of operating status. Error messages are displayed on the screen to indicate an improper operating condition or invalid entry on the command line. The softkey prompts are provided on the softkey label line to prompt the user to input the required information.

Table B-1. Status Messages

Status Message	Meaning
Awaiting command	Displayed when the software analyzer is in a quiescent state, ready to accept a new command in its command line.
Copy complete	Displayed when a *copy* command has been completed.
Copying	Displayed when the software analyzer is copying a display, setup, or measurement to a listing file or the system printer.
Database check, files = nn, errors = ee	Displayed during execution of the *database_check* command, where nn = the number of files checked and ee = the number of errors found.
Database search successful	Displayed when the analyzer has successfully completed the database search for the requested command.
Disable occurred	Displayed when the specified measurement disable term has been found and the measurement has been disabled (terminated).
Executing non-real-time	Displayed during execution of a measurement when *real_time opitonal* mode is selected (breaks to the emulation monitor may occur during execution of user program).

Executing real-time	Displayed during execution of a measurement when ***real_time required*** mode is selected (breaks to the emulation monitor are not allowed).
Executing (waiting for enable state # n)	Displayed after a measurement is started, but before all measurement enable terms are found. The software analyzer does not capture data until all measurement enable terns are found.
Executing (acquired = n)	Displayed during measurement execution. "n" is the number of states captured up to the current time.
Execution completed (saved = n)	This status message is displayed after a measurement has completed normally.
Execution halted (saved = n)	This message is displayed when a measurement is terminated by a ***Halt*** command.
Formatting display	Displayed while the acquired data is being formatted for display after completion of a measurement or after execution of a ***display*** command.
Formatting next page	Displayed while the next page of acquired data is being formatted for display after the (NEXT PAGE) key has been pressed.
Formatting previous page	Displayed while the previous page of acquired data is being formatted for display after the (PREV PAGE) key has been pressed.
Initiating cold start... Executing power-up sequence... Initializing 64340 hardware... Loading 64340 measurement software... Bootstrapping on-board processor... Initializing measurement data structures... Initiating emulation communication...	These status messages are displayed during the initialization of the software analyzer.
Loading configuration	Displayed when the software analyzer is in the process of configuring (either upon entry to the analyzer or during execution of a ***configuration load_from*** command).

Loading the hardware	Displayed after the ***execute*** command is given, while the analzyer hardware is being setup for the specified measurement.
Memory load complete	Displayed after execution of ***load*** command, indicating that the absolute file was successfully loaded into memory.
M68000--Reset	Indicates that the emulator was reset when last checked by the analyzer.
M68000--Running	Indicates that the emulator was executing the user program when last checked by the analyzer.
M68000--Running in monitor	Indicates that the emulator was running in the emulation monitor routine when last checked by the analyzer.
No stack information - dynamic variables cannot be displayed	Dynamic variables cannot be displayed when a measurement is started from a standing start from within a module or if a emulation break is recognized after the exit record occurs.
Save complete	Displayed upon completion of a ***configuration save_in*** command.
Saving configuration	Displayed when the software analyzer is in the process of saving a measurement configuration as a result of a ***configuration save_in*** command.
Searching database for file: <FILE>	Displayed when the analyzer is collecting database information for the named file.
Searching database for line: <LINE>	Displayed when the analyzer is collecting database information for the specified line or line range.
Searching database for module: <MODULE>	Displayed when the analyzer is collecting database information for the named module.
Searching database for variable: <VAR>	Displayed when the analyzer is collecting database information for the named variable.
Unloading acquisition memory (count = n)	Displayed while the data acquisition memory is being unloaded to the analyzer's on-board memory for postprocessing. "n" indicated the number of states unloaded.
Waiting for any keystroke	Displayed when a ***wait*** command has been executed. When used in a command file, ***wait*** suspends execution of the command file until a key is pressed on the HP 64000 keyboard.

Waiting for any keystroke or delaying n seconds	Displayed when a ***wait*** <SECONDS> command has been executed. When used in a command file, ***wait*** <SECONDS> suspends execution of the command file until a key is pressed on the HP 64000 keyboard or the specified number of seconds elapse.
Waiting for any keystroke or measurement complete	Displayed when a ***wait measurement_complete*** command has been executed. When used in a command file, ***wait measurement_complete*** suspends execution of the command file until a key is pressed on the HP 64000 keyboard or the current measurement is completed.
-- Window disable occurred --	Displayed in the trace list to indicate where a measurement window was disabled by the specified measurement_disable window term.

Table B-2. Error Messages

Error Message	Meaning
Access guarded mem.	Displayed when Memory is accessed that was not mapped in the emulation configuration file.
Access to variable: <VAR> not allowed at procedure	Indicates that the specified variable cannot be accessed at the requested procedure entry or exit point.
Address range must be <= 4K bytes	Displayed when a line range contains more that 4096 bytes in a ***count statements*** measurement.
Analyzer not in real time mode	Displayed if any IMB specifications are requested when the software analyzer has not been put in ***real_time required*** mode.
Bad line range	Indicates that the line numbers specified are at the same address or the second line number's address is less than the address of the first line number.
Boot failure for on-board processor	Indicates that the on-board processor failed to boot correctly. If repeated failures occur, execute ***option_test*** to verify that the analyzer is operating correctly.
Communication data overflow	Displayed if the on-board processor's communications with the host processor fail.
Configuration file corrupt	Indicates that the configuration file specified to be loaded is corrupted and cannot be used.
Counter mode may not be changed for this measurement	Indicates that the counter mode has been preset for the count or time measurement and cannot be changed using the ***setup counter*** command.
Duplicate module names in this file	Indicates that the file contains two modules with names that are identical in the first 15 characters. Duplicate module names are not supported by the software analyzer. The file must be modified to eliminate the duplicate names.
Duplicate module names specified: <MODULE>	Displayed when a module is specified multiple times in the setup specification. Modify the setup.
Emul_com file <FILE> is inconsistent with hardware	Displayed when the locations and/or board types in the station do not match the specification defined in the emulation command file

Fatal system error	Displayed if there is a fatal error in the software analyzer. Exit to the system monitor and re-enter the module. If repeated failures occur, execute ***option_test*** to verify that the analyzer is in working condition.
File exists, wrong module type	Indicates that a file of type "trace" with the specified name exists, but is not a software analyzer configuration file.
File is write protected	Indicates that the configuration file is write protected and cannot be overwritten with another configuration file. To remove the configuration file, you must purge or rename it.
File not found file= <FILE> :comp_db	Displayed when no comp_db file exists for the specified file. Relink absolute file using ***options comp_db***. Note: in C, this message appears at the call to "main" (e.g., in the command ***trace modules all...***). The module "entry" which calls "main" does not have a comp_db file. For this case, this message is normal and does not indicate an error condition.
File not found file= <FILE>:comp_db (PC= nnnnH)	Displayed when no comp_db file exists for <FILE>. <FILE> may be an assembly language file or a file for which no comp_db file was created. The PC address is the address within the file that caused the access to occur.
Hardware configuration error	A multiple module measurement execution command failed due to a hardware configuration error.
IMB drive/receive specifications must be turned off	All IMB specifications must be removed (***setup trigger_enable always***) before the software analyzer may be put in ***real_time optional*** mode.
IMB execution error	An error has occurred in an execution of a multiple module measurement.
IMB halt error -hardware error	Displayed if there was a hardware error as a result of a ***halt*** command in a multiple module measurement.
Incomplete multi-module specification	A multiple module measurement execution command failed due to an incomplete IMB specification.
Incorrect type for variable: <VAR>	Displayed when the symbol specified is not a variable type that can be used in the specified context such as specifying an array index on a pointer variable.

Invalid field for this measurement	Displayed when the display field selected is not valid for the current measurement.
Invalid symbol type encountered	Displayed when the database encounters a symbol type that is invalid.
Line not found: <LINE>	Indicates that the specified line has no code associated with it.
Line numbers are not in the same module	Indicates that the two line numbers specified are not contained in the same module.
Line range must be <= 255	Displayed when a line range contains more that 255 lines in a "setup count statements" measurement.
Local variables not allowed: <VAR>	Displayed when variables local to the procedure being traced in a trace data_flow measurement are requested. Local variables are not active on entry or exit and cannot be traced.
Main program illegal for this measurement: <MODULE>	The main program cannot be traced in a trace data_flow measurement.
Measurement cannot be accomplished real-time	Ths message is displayed when an attempt is made to execute a measurement in real-time mode that can only be executed in non-real-time mode. Redefine the measurement or select non-real-time mode.
Measurement data not available	Indicates that no measurement has been taken, and therefore a "show measurement" command may not be executed.
Module not found: <MODULE>	Displayed when the specified module cannot be found in the file indicated.
Monitor System Error: <ERROR>, Code: <CODE>	The software analyzer has had an internal software error. Please record <ERROR> and <CODE> to give to your HP representative.
Multiple drivers defined on trigger enable	Displayed if an IMB specification has been setup which results in multiple modules driving trigger enable. The error occurs when an ***execute*** is requested.
No absolute file defined	Displayed when an execute or run is requested with no absolute file defined. An absolute file may be defined by loading one in any of the 64000 modules, or be using the ***setup absolute_file <FILE>*** command.

No absolute file loaded	Displayed when a ***database_check*** command is requested when no absolute file has been loaded. Note: a ***setup absolute_file <FILE>*** command will not satisfy the requirements, a ***load*** command must be used.
No code generated for module: <MODULE>	Displayed when the module entry address is equal to the module exit address.
No file defined	Displayed if a ***show source*** command is requested with no file name given, and no default file defined.
No matching enable/disable defined, trigger_enable ignored	Displayed when a trigger enable has been defined without first setting up the measurement enable/disable that it is associated with. The condition is corrected once the measurement enable/disable is setup.
No modules in file: <FILE>	Indicates that the file specified to be traced contains no modules.
No module name found (PC= nnnnH)	Indicates that the data base, absolute file, or source file has been modified since the measurement was made. PC indicates that a program counter was executed which does not map to a module name.
No monitor program	Displayed when no emulation monitor program has been loaded in emulation memory. Relink your absolute file to include the emulation monitor program and reload the absolute file.
No path defined	Displayed when no path has been defined for a symbol. A symbol must have a path definition specified, either in the default path specification or as part of the symbol specification in the command line.
No source line found (PC= nnnnH)	Indicates that the PC is part of a source file but has no source line associated with it (e.g. the overhead generated by the compiler for procedure entries when the procedure is the first executable code in the file).
No transfer address - absolute file not loaded	Displayed when a ***run from transfer_address*** command is requested and no absolute file has been loaded.
On-board processor failed to respond	Displayed if the host processor's communications with the on-board processor fail.
Only one disable sequence state allowed non-real-time	More than one measurement disable term has be specified in non-real-time mode. Redefine the measurement disable specification.

On-board processor failure	Indicates a failure of the on-board processor to respond to the host system. ***option_test*** should be run to verify that the analyzer is in working condition.
Out of dynamic resources - simplify measurement	The specified measurement requires more dynamic resources than are available in the software analyzer. Redefine the measurement so that it requires fewer dynamic resources.
Out of high level resources - simplify measurement	The specified measurement requires more high level resources than are available in the software analyzer. Redefine the measurement.
Out of low level resources - simplify measurement	The specified measurement requires more low level resources Than are available in the software analyzre. Redefine the measurement.
Processor not in monitor	An attempt has been made to execute a command requiring the emultion monitor to be running while the user program is executing. Break the processor before attempting to execute the command.
Processor not supported	Indicates that the software analyzer does not support the processor for which the current emulation command file is set up, or that the software analyzer software for that processor is not loaded on your system disc.
Program execution outside of absolute file (PC= nnnnH)	Displayed when the emulator is executing code outside of the address space defined for the absolute file.
Real time required; specify run at_execution from emulator	Displayed when the analyzer is in a real_time required mode and ***run at_execution*** is requested. The action may be simulated by executing the measurement with the emulator not yet running the user program, exiting the software analyzer module, and then entering emulation and executing a run there.
Run at execution must be removed	A pending ***run at_execution*** command must be removed before the software analyzer may be put in a real_time required mode.
Setup specification overflow, simplify request	Indicates that the setup command requested is larger than the analyzer can accept. This is especially possible with the ***all*** specification on files that have a large number of procedures.

Specify run at_execution from emulator for IMB measurements	An attempt has been made to specify a run at_execution command from within the software analyzer when it is configured for an IMB measurement. When making IMB measurements, you must specify run at_execution from the emulator.
Sub-element not found for variable: <VAR>	Displayed when the variable expression specified is not found for the variable specified.
Symbol is too big, modify sub-elements individually	Displayed when the requested variable is greater than 4 bytes. Modify variable in smaller units.
Symbol(s) type information is too big	Indicates that the type information describing the symbol(s) to be traced or displayed exceeds the maximum size allowed.
Syntax invalid	Displayed when an attempt is made to execute a syntactically invalid command.
Too many symbols specified	Indicates that more than the maximum number of symbols are specified in the setup.
Type information too big for variable: <VAR>	Indicates that the type information describing the symbol to be used exceeds the maximum size allowed.
Unable to access monitor	The software analyzer is unable to access the emulation monitor program. Verify that the emulation monitor is loaded to emulation memory.
Undefined size for variable: <VAR>	The variable is a C array with unspecified size and cannot be traced.
Unsupported type for variable: <VAR>	Indicates that the specified variable's type is not supported by the software analyzer.
Variable access not allowed at procedure for: <VAR>	Indicates that the specified variable cannot be accessed at the requested procedure.
Variable access not allowed from current PC	Indicates that the variable is a local variable and is not scoped to this location in the program.
Variable expression has too many indirections	The variable expression exceeded the maximum number of indirections supported by the software analyzer.

Variable not found: <VAR>	Displayed when the specified variable cannot be found in the procedure and/or file indicated.
Warning: multiple drivers defined on trigger enable	Displayed if an IMB specification has been setup which results in multiple modules driving trigger enable. Execution of a measurement will not be allowed as long as this condition exists.
Warning: value parameters will not be displayed on module exit	Indicates that a value parameter was requested in a trace data_flow measurement on a procedure exit. The variable is not active on procedure exit and cannot be displayed.

Table B-3. Softkey Prompt Messages

Softkey Prompt	Message and Meaning
<ADDRESS>	*Any address constant* <ADDRESS> is any valid address within the absolute file loaded into user or emulation memory.
<CMD_FILE>	*A command file name* <CMDFILE> prompts the user to enter the name of a command file containing valid software analyzer commands to automatically configure the analyzer or execute a measurement.
<FILE>	*Filename[:Userid][:Disc#]* When used with the "load" command, <FILE> is the name of the absolute file to be loaded from the 64000 system memory in user RAM or emulation memory.
<FILE>	*Filename[:Userid][:Disc#]* When used with the "copy" command or "database_check" command, <FILE> is the name of the listing file that the display, setup, measurement, or database information is to be copied to.
<FILE>	*A source file name. Note: ':' may replace 'file'* <FILE> is an optional parameter that refers to the source file containing the specified <MODULE>, <VAR>, <PROC>, <LINE>, or line range specified in the command statement. If the <MODULE>, <VAR>, <PROC>, <LINE>, or line range is in the defined default path, the <FILE> parameter may be omitted from the command statement.
<INDEX>	*An index value or scalar* An index value (integer or scalar value) specifying a component of an array.
<INVALID>	*Command syntax is invalid* The portion of the command between the beginning of the command and the cursor contains errors in syntax and must be corrected before the command may be entered.

<LINE> *A program line number*

<LINE> represents the line number of a Pascal or C statement in the source program. If the specified <LINE> contains only comments (no executable code), the analyzer will associate the line number with the first line containing executable code following it. Any comment lines preceding the first line of executable code in a procedure or function are not recognized by the software analyzer. All lines in the specified line range must be contained within a single modules. This module may be a procedure or function in Pascal or a function in C, or the main program block.

<MODULE> *A program or module name*

<MODULE> represents the name of a contiguous segment of code with a single entry point and a single exit point. In Pascal, a module can be the name of a procedure function or the main program within a specified file. In C, a module can be the name of a function within a specified file. If a module name is the same as a software analyzer keyword, e.g. entry, you must enclose the module name in quotes ("entry") in the command statements.

<PARMS> *Command file parameters*

The parameters passed to a command file.

<PROC> *A procedure name. Note: '@' may replace 'proc'*

<PROC> is an optional parameter that refers to a procedure or function. If <PROC> is defined in the default path, it may be omitted in the command line. If <PROC> is not specified in either the default path or the command, the analyzer assumes that <VAR> is a global variable defined at the main program level.

<RETURN> *Command syntax is valid to cursor*

The portion of the command between the beginning of the command and the cursor contains no errors in syntax and can be entered if no further options are desired.

<SECONDS> *A number of seconds to delay*

Used with the "wait" command to specify the number of seconds to pause before again accepting commands.

<STATE #> *An integer value*

A positive integer value within the range 0 thru 9999 used to select a position in the measurement data for copy or display. The value specifies the number of bytes offset from the start of the measurement data.

<SYMBOL> *A valid global or local (specify file) symbol*

<SYMBOL> allows the user to specify program execution to run from a specified symbol. If a file name is specified with <SYMBOL>, the analyzer assumes that the symbol is a module in the specified file. If no file is specified with <SYMBOL>, the analyzer first looks for the address of a global symbol in the link_sym file associated with the currently loaded absolute file. If no global symbol is found there, the analyzer then searches for a module in the current default file.

<VALUE> *An integer value (32 bits or less)*

<VALUE> represents the value that the specified variable is to be changed to in a "modify <VAR>" command. <VALUE> must be specified as an integer value.

<VAR> *A valid variable identifier with pointers*

Used with the "display <VAR>", "modify <VAR>", and "setup trace data_flow" commands. <VAR> represents the name of a valid program variable, local variable, or parameter. <VAR> can be any valid Pascal or C variable expression, including pointers. If a variable name is the same as a software analyzer keyword, e.g. entry, you must enclose the variable name in quotes ("entry") in the command statements.

<VAR> *A valid variable identifier*

Used with the "setup trace variables", "setup measurement_enable access <VAR>", and "setup measurement_disable access <VAR>" commands. <VAR> represents the name of a valid program variable, local variable, or parameter. <VAR> can be any valid Pascal or C variable expression. No pointers are allowed. If a variable name is the same as a software analyzer keyword, e.g. entry, you must enclose the variable name in quotes ("entry") in the command statements.

<WIDTH> *An integer width from 0 to 132 columns*

An integer value from 0 to 132 used with the "display" command to specify the width in columns of the specified field.

Appendix C

STACK ARCHITECTURE AND MEMORY STRUCTURE

INTRODUCTION

In order for high level software analysis to be performed on more than one processor it is necessary to tailor the analyzer software to the particular processor. Processor specific capabilities are needed for several reasons. Primarily, since most processors have different architectures and instruction sets, compiler designers will often define the stack architecture for their compiler based on the characteristics of the processor the compiler is targeted for. Consequently, the information required for high level analysis that must be obtained from the stack is not the same from processor to processor. Additionally, the location of compiler generated code used to build the stack varies from compiler to compiler based on the compiler designer's decisions. This affects the way in which the stack reference point is established for analysis, i.e., if the stack is built prior to calling a procedure, upon entry to the procedure the stack pointer is located in a different relative position on the stack than if the stack is built after entry to the procedure.

In addition to the stack architecture, physical memory allocation can vary from one processor to the next making it necessary to be able to accurately interpret the basic structure of the memory. These requirements indicate the need for software personality modules designed to interpret the stack structure of each compiler so that location of parameters, variables and parental information is readily obtainable during analysis of program execution as well as to interpret the physical memory allocation for the processor.

STACK ARCHITECTURE

Pascal Compiler Considerations

The stack architecture for the Pascal compiler is illustrated by the stack frame diagram in figure C-1. A stack frame is created for each procedure call during program execution. Register A6 is used as the base pointer for the stack. Register A7 is the stack pointer. The base pointer is established prior to subroutine calls and is used directly as the reference point for obtaining the needed information from the stack. As the data base is being built during link time, stack addressing information is entered in the data base as an offset from register A6. During run time, when the procedure or function is entered, a software break, set up by the software analyzer, occurs and register A6 is read and saved. In this manner, the stack information required by the measurement may be addressed by adding the offset from the data base to the address contained in register A6.

The static link pointer is created only if the procedure described by the stack frame is nested (level 2 or greater). The static link points to the parent routine's static link. If a parameter is passed by reference, its 4-byte address is placed on the stack in the parameter field. If the parameter is passed by value and is shorter than or equal to 4 bytes, its value is placed in the parameter field. If a value parameter is longer than 4 bytes in size, it is placed in the large value parameter field. If a function returns a value longer than 4 bytes in size, the address of the value is placed in the optional function return value address field. Otherwise the value is returned in a register and the field is not created. The variable 1 through variable N fields contain local variables. The previous frame

pointer points to the stack frame of the calling routine. The temporary storage buffer is used by the compiler.

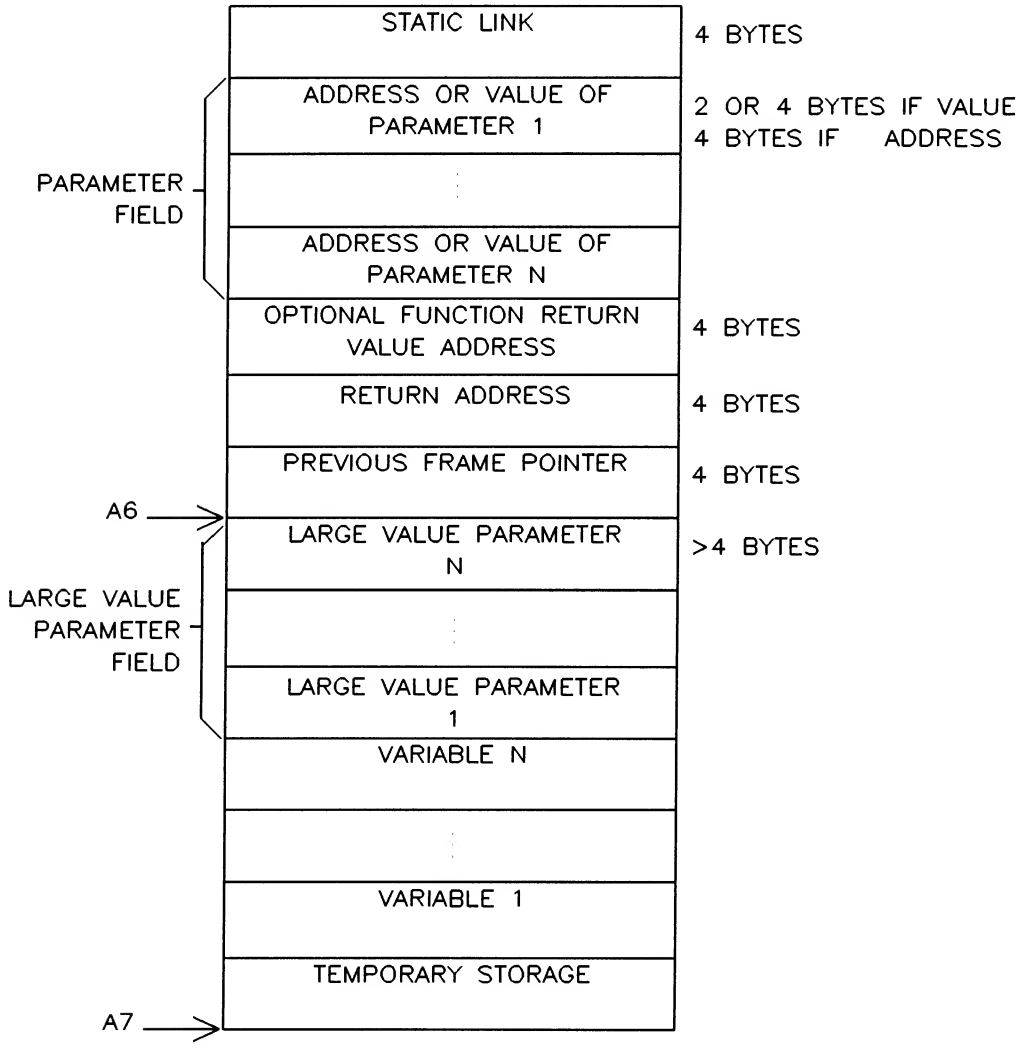


Figure C-1. Pascal Stack Frame

C Compiler Considerations

All parameters in C programs are passed by value, except arrays, which are effectively passed by reference. If arrays are passed as parameters with an unspecified length, they will be unbounded in the compiler symbol table. Consequently, the software analyzer will be unable to trace the parameter "A" if "A" is an array of unspecified length. In order for the software analyzer to trace an unbounded array passed as a parameter, the user must specify a specific element of the array "A[N]" where N specifies an element of an array of unspecified length.

The stack frame structure for the C compiler is very similar to that of the Pascal stack frame. With the FIXED_PARAMETERS compiler option ON (figure C-2), the stack frame structure is identical except that, since C does not permit nested functions, no static link field is created. If the FIXED_PARAMETERS option is OFF (figure C-3), the order in which parameters are placed on the stack is reversed.

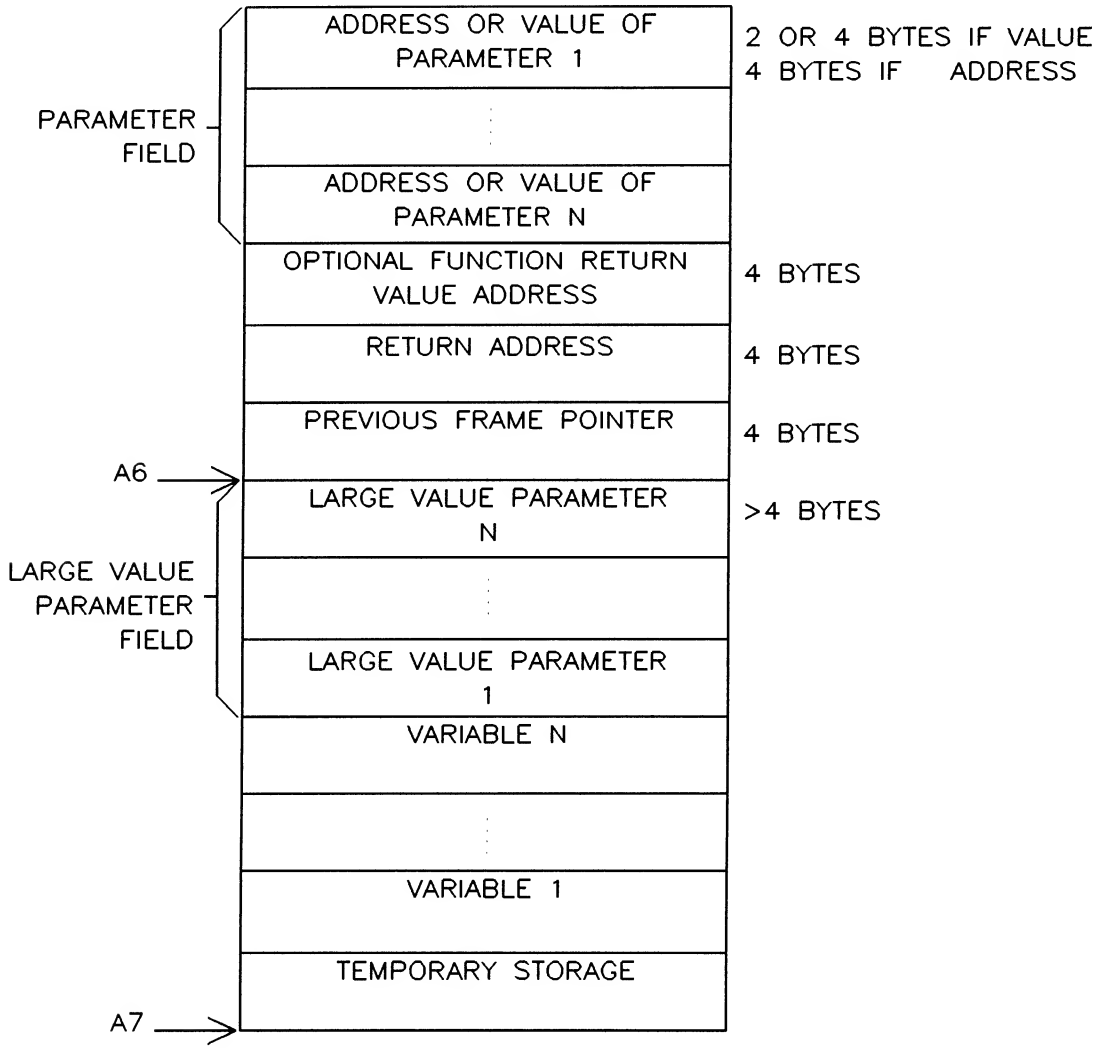


Figure C-2. C Stack Frame (Fixed Parameters Options On)

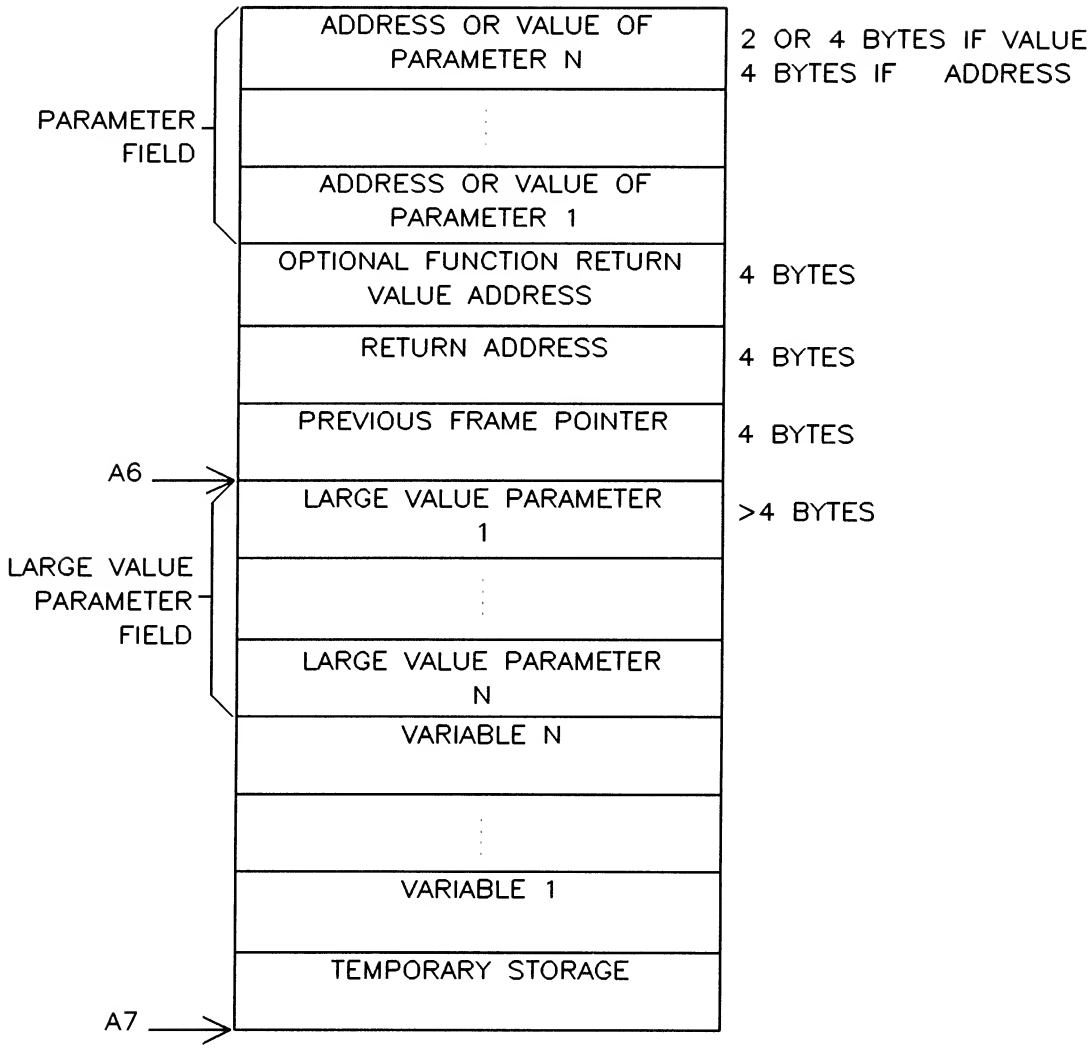


Figure C-3. C Stack Frame (Fixed Parameters Options Off)

Appendix D

GLOSSARY OF SOFTKEY LABELS

INTRODUCTION

Table D-1 contains a list of the softkey labels provided in the software analyzer software. The corresponding command line message is given for each softkey label and an explanation of the softkey label follows. An example is also given which shows the message as it would appear on the command line.

Table D-1. Software Analyzer Softkey Labels

Softkey Label	*Command Line Message*

absolute *absolute*

Used to define that counts or times should be displayed absolute with respect to the beginning of the trace.

display count absolute

absolute *absolute_file*

Used to define an absolute file name in the event that none was loaded via any 64000 analysis subsystem.

setup absolute_file NT1:TEST

address *address*

Used with the *run* command to indicate that the information that follows is an address constant specified in binary, octal, decimal, or hexadecimal.

run from address 2476H
run at_execution from address 3744Q

all *all*

Used with the *setup trace modules* command to specify that all modules in a file will be traced. If no file is specified in the command, all modules in the default file are traced.

setup trace modules all

always *always*

Used to define IMB (intermodule bus) as being always enabled. This has the effect of taking the analyzer off the IMB.

setup trigger_enable always

any_state *any_state*

Used in setting up measurement enables/disables as a condition which is true on any_state encountered.

setup measurement_enable on any_state

append *append*

Used with the *copy* and *database_check* commands to append information to an existing listing file.

copy display to DSPL *append*
database_check listfile DBC *append*

ascii *ascii*

Used with the *display base* command to display values in the measurement display value field as ASCII characters.

display base ascii

at_exec *at_execution*

Used with the *run* command to start running the user's software in emulation from a specified location at execution of a trace measurement.

run at_execution from PROC2

binary *binary*

Used with the *display base* command to display values in the measurement display value field as binary numbers.

display base binary

break *break*

Used in three ways:

1. To define a break measurement.

   setup break on 123 *or* PROC2

2. To define whether or not to break the user's program on completion of a measurement.

   setup break off
   setup break on measurement_complete

3. To break the user's program forcing the emulator back into the monitor.

   break

cnt_state *to_count_states*

Used in the *setup counter* command to specify that the hardware counter is to count bus cycles.

setup counter to_count_state

cnt_time *to_count_time*

Used in the *setup set_counter* command to specify that the hardware counter is to time.

setup counter to_count_time

configure *configuration*

Used to save the analyzer configuration in, or load the analyzer configuration from a file. The file type is trace and contains the entire configuration for the analyzer.

configuration load_from SETUP:USER
configuration save_in SETUP:USER *protected*

copy *copy*

Used to copy the setup, measurement, or the current display to a listing file or to the printer.

copy measurement to printer
copy setup to LISTFILE

count *count*

count_statements

Used in two ways:

1. To define a count statements measurement.

setup count_statements PROC2

2. To define the mode to display count/time information on the display. The choices are relative, absolute, and statistics.

display count relative

counter *counter*

Used to define whether the hardware counter is to count time or count bus cycles during the execution of a measurement. The default value is count time.

setup counter to_count_time

data_flow *data_flow*

Used with the *setup trace* command to specify the *trace data_flow* measurement.

setup trace data_flow PROC2 (X , Y , Z)

db_check *database_check*

Used to specify a database compatibility check.

database_check listfile display

decimal *decimal*

Used with the *display base* command to display values in the measurement display value field as decimal numbers.

display base decimal

default *default*

Used with the *display* command to display a measurement in the default format or with the *display base* command to display the value field of the measurement in the default base.

display default
display base default

dflt_path *default_path*

Used with the *setup* command to define a default path (a module within a file or a file itself). The default path is used when a path is needed for a measurement and is not included in the measurement command itself.

setup default_path proc PROC1 *file* NT1:TEST

disable *measurement_disable*

Used in five ways:

1. In the *setup* command to define the condition which, if found, will result in the termination of the measurement.

setup measurement_disable on 123 *or* PROC1

2. In the *setup* command to remove a defined disable condition.

setup measurement_disable off

3. In the *setup trigger_enable received* command to specify that trigger enable should be received on format ment disable.

setup trigger_enable received measurement_disable

4. In the *setup trigger_enable driven_only* command to specify that trigger enable should be driven on measurement disable.

setup trigger_enable driven_only measurement_disable

5. In the *setup modify* command to modify the current definition of the measurement_disable condition.

setup modify measurement_disable

display *display*

Used in four ways:

1. To display the value of variables in a specified procedure and/or file. If no procedure or file is specified in the command, the software searches for the variable(s) in the default path.

display A[3] *proc* PROC1 *file* AVER

2. To selectively format and display fields in the measurement display (default, modify, source, source_path, value, symbol, symbol_path, count, status).

display value then symbol then symbol_path

3. To copy a display to a file or to the system printer.

   copy display to printer

4. To list the database_check results to the display.

   database_check listfile display

driven *driven_only*

Used in the IMB specification to define whether trigger enable is to be driven on measurement enable or measurement disable.

setup trigger_enable driven_only measurement_enable

emulation *emulation_memory*

Used with the *load* command to load absolute code from the 64000 system disc into emulation memory. The destination of the absolute code is determined by the address specified during linking.

load emulation_memory AVER

enable *measurement_enable*

Used in five ways:

1. In the *setup* command to define the condition which, if found, will enable the software analyzer to execute the specified measurement.

   setup measurement_enable on 123 *or* PROC1

2. In the *setup* command to remove a defined enable condition.

   setup measurement_enable off

3. In the *setup trigger_enable received* command to specify that trigger enable should be received on measurement enable.

   setup trigger_enable received measurement_enable

4. In the *setup trigger_enable driven_only* command to specify that trigger enable should be driven on measurement enable.

   setup trigger_enable driven_only measurement_enable

5. In the *setup modify* command to modify the measurement_enable specification.

   setup modify measurement_enable

end *end*

Used in two ways:

1. With the *copy measurement* command to copy the measurement from the current trace data location (indicated on the status line) through the end of the measurement data.

copy measurement thru end to printer

2. To end a software analysis session.

end

entry *entry*

Used in two ways:

1. With the *setup trace data_flow* command to specify that variables be traced only on entry to the specified module(s). The default condition is to trace variables on both entry to and exit from a module.

setup trace data_flow PROC1 *entry* (A,B)

2. With the *setup measurement_enable* and *setup measurement_disable* commands to specify that the measurement be enabled or disabled on entry to the specified module.

setup measurement_enable on PROC2 *entry*

execute *execute*

Starts execution of a measurement. The *repetitive* option causes the measurement to be repetitively executed.

execute
execute repetitive

exit *exit*

Used in two ways:

1. With the *setup trace data_flow* command to specify that variables be traced only on exit from the specified module(s). The default condition is to trace variables on both entry to and exit from a module.

setup trace data_flow PROC2 *exit* (A,B)

2. With the *setup measurement_enable* and *setup measurement_disable* commands to specify that the measurement be enabled or disabled on exit from the specified module.

setup measurement_disable on SORT:BUB_SORT *exit*

file *file*

Used to indicate that the name of a source file follows. **NOTE:** A colon (:) may be used in place of pressing the *file* softkey.

setup default_path file TESTP
display SAM *proc* PROC4:NT1:TEST

followed *followed_by*

Used with the *setup measurement_enable* and *setup measurement_disable* commands to specify sequential enable or disable conditions.

setup measurement_enable on PROC1 *followed_by* SORT

from *from*

Used with the *run* command to specify the location in the user's program from which program execution will begin in emulation.

run from transfer_address

halt *halt*

Used to halt the measurement currently in process, or to halt the unloading of the data acquisition memory. The data collected before the *halt* command was executed is displayed.

halt

hex *hex*

Used with the *display base* command to display values in the measurement display value field as hexadecimal numbers.

display base hex

listfile *listfile*

Used with the *database_check* command to select one of three output devices for listing results; the display, the printer, or a file.

database_check listfile RESULTS *append*

load *load*

Used to load absolute files from the 64000 system disc into user RAM or emulation memory.

load emulation_memory TESTP

load_from *load_from*

Used with the *configuration* command to configure the software analyzer as specified in the configuration file being loaded. The configuration file is of type trace.

configuration load_from SETUP:USER

measure *measurement*

Used in three ways:

1. In the *setup modify* command to modify the current measurement definition.

   setup modify measurement

2. In the *show* command to show the current measurement data.

   show measurement

3. In the *copy* command to copy the current measurement data to the printer or to a file.

   copy measurement to RESULTS *append*

meas_comp *measurement_complete*

Used in two ways:

1. With the *setup break on* command to break the user's program when a measurement is completed.

   setup break on measurement_complete

2. With the *wait* command to suspend execution of a command file until the current measurement is completed.

   wait measurement_complete

modify *modify*

Used in three ways:

1. To modify the current value of a variable in emulation or user memory.

   modify Q.CHAR1 *proc* LTRSORT = 41H

2. With the ***display*** command to modify the current display definition. The ***display modify*** command recalls the current display definition to the command line for editing, eliminating the need to re-enter an entire display command.

display modify

3. With the ***setup*** command to modify the current measurement, measurement_enable, or measurement_disable definition.

setup modify measurement_enable

modules ***modules***

Used with the ***setup trace*** command to specify that modules (functions or procedures) are to be traced.

setup trace modules all

octal ***octal***

Used with the ***display base*** command to display values in the measurement display value field as octal numbers.

display base octal

off ***off***

Used in four ways:

1. In the ***setup break*** command to turn off a break on measurement complete.

setup break off

2. In the ***setup measurement_disable*** command to turn it off.

setup measurement_disable off

3. In the ***setup measurement_enable*** command to turn it off.

setup measurement_enable off

4. In the ***run*** command to turn off the ***at_execution*** parameter.

run at_execution off

on *on*

Used in four ways:

1. In the *setup break* measurement to define the conditions on which to break.

   setup break on 123 *or* PROC1

2. In the *setup break on measurement_complete* command.

   setup break on measurement_complete

3. In the *setup measurement_disable* command to define the conditions on which to disable.

   setup measurement_disable on 123 *or* PROC1

4. In the *setup measurement_enable* command to define the conditions on which to enable.

   setup measurement_enable on 123 *or* PROC1

optional *optional*

Used in the *setup real_time* command to enable the analyzer to break to the emulation monitor.

setup real_time optional

or *or*

Used as a logical combinatoric for inclusive ORing of conditions for the *setup break on* measurement, and *setup measurement_disable on* and *setup measurement_enable on* conditionals.

setup break on 123 *or* PROC1
setup measurement_enable on PROC2 *or* 115

printer *printer*

Used in two ways:

1. In the *copy* command to specify that the display, setup, or measurement be copied to the system printer.

   copy display to printer

2. In the *database_check* command to specify that the results be copied to the system printer.

   database_check listfile printer

proc *proc*

Used to indicate that a procedure or function name follows that defines the procedure or function to which a variable belongs. NOTE: an "@" may be used in place of *proc*.

setup trace variables FLAG1 *proc* FUNCTIONX

protected *write_protected*

Used with the *configuration save_in <FILE>* command to prevent the accidental modification of the file with a later *configuration save_in* command. The file is protected against writes only within the software analyzer. It can still be purged, renamed, or copied into from the system monitor level.

configuration save_in SETUP:USER *write_protected*

read *read*

Used to specify that only memory read accesses to a variable be traced. The default condition is to trace both memory read and memory write operations on a specified variable.

setup trace variables QFLAG *read*

real_time *real_time*

Used in the *setup* command to specify whether the analyzer is allowed to break to the emulation monitor during a measurement.

setup real_time required

received *received*

Used in the *setup trigger_enable* command to define that trigger_enable is to be tied to either measurmement disable or measurement enable.

setup trigger_enable received measurement_enable

relative *relative*

Used to define that counts or times should be displayed relative with respect to the proceeding count or time in the trace.

display count relative

repeat *repetitive*

Used in the *execute* command to specify that trace measurements are to be executed repetitively until a *halt* command is issued.

execute repetitive

required *required*

Used in the *setup real_time* command to define that the analyzer may NOT break to the emulation monitor.

setup real_time required

reset *reset*

Used to suspend emulation system operation and reestablish initial operating parameters. The reset signal is latched when active and is released by the *run* command.

reset

run *run*

If the processor is in a reset state, *run* will cause the reset to be released and, if a *from* address is specified, the processor will begin program execution at that address. If the processor is running in the emulation monitor, the *run* command causes the processor to exit into the user program.

run from address 312EH

save_in *save_in*

Used with the *configuration* command to save the software analyzer configuration in a file. This file is of type trace.

configuration save_in SETUP:USER

setup *setup*

Used in three ways:

1. To specify the measurement parameters and conditions with which the software analyzer will run. These include *trace*, *count*, *time*, *break*, *measurement_enable*, *measurement_disable*, *counter*, *real_time*, *default path*, and *trigger_enable*.

setup default_path NT1:TEST
setup trace modules PROC1 , PROC2
setup measurement_enable on 123H
setup real_time optional

2. In the ***show*** command to display the software analyzer setup.

   show setup

3. In the ***copy*** command to copy the measurement setup to the system printer or to a file.

   copy setup to printer

show ***show***

Used to specify what information is to be displayed. You may show the software analyzer setup, the current measurement data, or a source file.

show measurement
show setup
show source NT1:TEST

source ***source***

Used in two ways:

1. In the ***show*** command to display a source file. If no source file is entered, the default path file is used.

   show source NT1:TEST

2. In the ***display*** command to display the source field in the measurement trace listing.

   display source

src_path ***source_path***

Used in the ***display*** command to display the source_path field in the measurement trace listing showing the source file name that the source statement was extracted from.

display source_path

start ***start***

Used with the ***copy measurement*** command to specify that measurement results from the start of the trace to the current line be copied to a file or the system printer.

copy measurement thru start to LIST

statement *statements*

Used in the *setup trace* command to set the software analyzer measurement mode to *trace statements*.

setup trace statements PROC2

statistic *statistics*

Used in the *display* command to display the statistics field (minimum, maximum, mean, and count) in the trace listing on the measurement display. This field is only valid for a *time modules* measurement.

display count statistics

status *status*

Used in the *display* command to display the status field in the measurement trace listing.

display value then status

sw_anl_N *sw_anl_N*

Used in the measurement system level of softkeys to enter the software analyzer. May be followed by an optional software analyzer configuration file name specifying a configuration file from which the analyzer is to be configured from or an emulation command file name of file type emul_com. *N* is the number of the 64000 card slot containing the software analyzer CPU board.

sw_anl_6 SETUP_1

symbol *symbol*

Used in the *display* command to display the symbol field in the measurement trace listing.

display symbol then status then source

symb_path *symbol_path*

Used in the *display* command to display the symbol_path field in the trace list, showing the path in which the symbol is defined. For modules, the symbol_path contains a file name. For variables and parameters, the symbol_path may be a file or a module and file, depending upon the level at which the symbol is defined.

display symbol then symbol_path

then *then*

Used as a delimiter to separate sequential field specifications in the ***display*** command.

display source then symbol then status

thru *thru*

Used with the ***copy measurement*** command to specify a range of data in the trace listing to be copied to a file or to the system printer. The minimum amount of data copied is the contents of the current display.

copy measurement thru end to printer

time *time_modules*

Used in the ***setup*** command to specify a time modules measurement.

setup time_modules PROC1 , PROC2

to *to*

Used in three ways:

1. With the ***setup trace statements*** command to specify a line range to be traced in a source program. All lines in the specified range must be contained in a single module.

setup trace statements 57 ***to*** 86

2. In the ***setup count_statements*** command to to specify a line range to be counted in a source program. All lines in the specified range must be contained in a single module. The total number of lines must not exceed 255, and the total address space the range covers must not exceed 4096.

setup count_statements 57 ***to*** 86

3. With the ***copy*** command to specify either a listing file or the system printer.

copy display to printer

trace *trace*

Used with the ***setup*** command to specify the trace measurement mode to be executed by the software analyzer. The measurement modes are ***trace data_flow***, ***trace modules***, ***trace statements***, and ***trace variables***.

setup trace statements PROCEDURE1 ***file*** TESTP

transfer *transfer_address*

Used with the *run* command to specify that the emulator begin program execution at the address stored in the transfer buffer (XFR_BUF). This is the starting address of the user program.

run at_execution from transfer_address

trig_enab *trigger_enable*

Used with the *setup* command to define the IMB interaction. Options are *always*, *received*, and *driven_only*.

setup trigger_enable driven_only measurement_enable

user *user_memory*

Used with the *load* command to specify that the absolute program be loaded into user RAM in the target system.

load user_memory FILENAME:USER

value *value*

Used in the *display* command to display the value field in the measurement trace listing.

display value

variables *variables*

Used with the *setup trace* command to specify that the software analyzer operate in the trace variables mode.

setup trace variables A , B , C

wait *wait*

Causes execution of a command file to be suspended until the current measurement being executed is completed. *measurement_complete* option must be used with the *wait* command.

wait measurement_complete

write *write*

Used with the "setup trace variables" command to specify that only memory write accesses to the variable be traced. The default condition is to traace both memory read and memory write operations on a specified variable.

setup trace variables QFLAG *write*

NOTES

Appendix E

RESOLVING MEASUREMENT PROBLEMS

INTRODUCTION

This appendix describes measurement problems you may encounter while using the software analyzer, their possible causes, and suggested solutions. Measurement abnormalities may result from the use of certain compiler directives, improper use of compiler and linker options, use of breaks, how measurements are implemented in the software analyzer, and many other causes. This appendix lists the most common problems, their causes, and suggested solutions to the problem. Programming style can also affect how the analyzer traces data. The section, ***Recommended Programming Style***, in chapter 3 gives guidelines for writing code to achieve the best results from your software analyzer.

MEASUREMENT PROBLEMS AND SOLUTIONS

Missing Source Statements

Problem: An expected "go to" statement is not displayed in trace statements. The analyzer may miss a "go to" statement if a multiword access is on the preceding source line.

Solution: Execution of the "go to" is seen in the change of source line numbers. To see the "go to" statement, structure the code so that a multiword access is not on the preceding source line.

Problem: A statement containing an "end" (in Pascal) or "}" (in C) is not displayed. The analyzer uses the "end" or "}" to indicate the end of user code. This can cause unexpected results in the trace display.

Solution: Place the "end" statement of a procedure in Pascal or the "}" terminator in a C function on a separate line containing no code.

Problem: After tracing statements in ***real_time required*** mode, source lines between data accesses disappear when trying to position measurement data on the screen by specifying a state number. Specified state is within a multibyte variable access and is not the first state of the variable access.

Solution: Use display positioning (roll keys, page keys, specifying state numbers) to find a location in the trace data that causes the source lines to reappear. Repetitively decrement the state number by 2 until source lines appear. Alternatively, try rolling the center display line off the screen.

Problem: When executing trace statement "don't care", only emulation monitor code is acquired.

Solution: (1) Select ***real_time required*** mode. (2) Do not use ***at_execution*** parameter in ***run*** command. (3) Do not specify ***setup measurement_enable*** on any_state.

Problem: When tracing statements with a single line specified, only one source line appears although the line is executed more than once.

Solution: Make sure that the measurement display includes additional fields along with the ***source*** field, e.g., ***count*** and/or ***value*** field.

Problem: Missing data during recursive calls in trace modules. Data is not displayed during recursive calls in trace modules when the recursive level is so deep that indentation causes the module not to appear on the display.

Solution: Expand the screen width of the symbol field with the "display" command. In some cases, the number of recursive levels may cause enough indentation that the maximum field width may be exceeded and the data cannot be shown.

Problem: In the trace variables measurement, the source line is missing for the entry of the first procedure in a file when large value parameters are being passed and the variable is being traced. The compiler overhead for entry into the procedure is executing. The source line is undefined for these instructions.

Solution: None.

Problem: Source lines are not displayed when, in non-real time mode, a break is executed within a procedure and then a trace statements measurement is executed from a standing start, i.e., measurement starts within a module with ***run at_execution*** (from next_pc) specified. Non-real time trace statements measurements from a standing start can cause missing source lines.

Solution: None.

Missing Symbols On The Display

Problem: The software analyzer cannot display a variable maintained in a register.

Solution: Turn the compiler option AMNESIA on.

Problem: The software analyzer cannot display an array parameter in C without an explicitly defined size.

Solution: Declare the maximum size required for the array.

Problem: The software analyzer cannot display the object of a pointer in the trace statements and trace variables measurements.

Solution: Break the processor in locations where you want to look at the pointer and use the display command. The trace data flow measurement can also be used trace the object of a pointer.

Problem: The software analyzer only displays the first specified variable of differently named variables mapped to the same location.

Solution: When setting up a measurement, specify first the more important variables that you want to see.

Problem: If a break is executed within a procedure and then a trace variables measurement is executed from a standing start (run at exection from next pc), all dynamic variable accesses are lost until an entry point to the procedure is detected.

Solution: The entry point of the procedure must be seen by the analyzer. Run the program from the point just before the recursive calls begin. Alternatively, declare the stack in the source program and trace the stack. Type information will not be available but the values will be displayed.

Problem: Function return values are not displayed.

Solution: The assignments to variables that receive function return values can be traced with the trace statements or trace variables measurement. Using a temporary variable within the function to compute the function value may be a useful way to trace function return values.

Problem: In a trace statements measurement, accesses to variables by the last instructions executed within the address range do not appear with prefetched processors.

Solution: If you are tracing a line range, add a couple of lines to the end of the range. If you are tracing a module, adding a dummy assignment statement at the end of the procedure will solve the problem.

Problem: A variable accessed by a source statement containing a type conversion is not displayed. The type conversion was accomplished by a call to a library routine and cannot be displayed.

Solution: None.

Unexpected Analyzer Execution

Problem: The analyzer doesn't capture any data. The measurement may be set up incorrectly.

Solution: Check ***run*** parameters. Try resetting the emulator and reloading the file.

Problem: The comp_db file is not current with the absolute file.

Solution: Run the db_check and correct the indicated errors.

Problem: When running in real time required mode, the analyzer stops acquiring data, but the measurement does not complete. The program has halted or an emulation error has occurred.

Solution: Use the ***halt*** command to stop the analyzer and view the measurement results. Run the program in non-real time to see why the program halted and to see emulation status and error messages.

Problem: A local variable cannot be modified. The variable is not scoped to the current program counter.

Solution: Execute a break in the code where it is valid to access the variable, run until the break, then modify the variable.

Problem: Real, character, or scalar variables can not be modified in their native type. Only variables that are 32 bits or less can be modified and only numerically.

Solution: None.

Problem: When executing a trace statements measurement in real-time required mode, no dynamic variables are displayed. In real-time required mode, the analyzer has to access to stack information.

Solution: Define an external array (static variable) that maps to the stack space.

Unexpected Emulation Operation

Problem: An error occurs on execution of a measurement after an absolute file containing the emulation monitor program is loaded and the emulator is currently running in the monitor. A load cannot be done correctly while code is running.

Solution: The processor must be reset before loading an absolute file containing the emulation monitor program.

Unexpected Error Or Status Message.

Problem: In ***real_time optional*** mode, The program runs until completion in the emulator and in the software analyzer the message "***running***" is the only one given. The software analyzer only checks the status of the emulator during an ***execute*** command, a ***show setup*** command, or on execution of a ***reset***, ***break***, or ***run*** command.

Solution: Perform one of the above operations to update the status of the emulator.

Problem: The program being traced crashes, the software analyzer gives only the error message "*access to guarded memory*", and no trace data is displayed. Something is fundamentally wrong with the program. No source code is able to execute. For example, the stack may have been placed in a section of memory that was guarded.

Solution: Exit the software analyzer and use the emulation subsystem to discover the problem. Assembly language tracing is required.

Problem: An absolute file was loaded without the emulator being reset.

Solution: Reset the emulator, then reload the absolute file.

Problem: The error message "*access to guarded memory*" appears on execute. The comp_db file is not current with the absolute file.

Solution: Run the db_check and correct the indicated errors.

Problem: The error message "*File not found file= <FILE>:comp_db (PC=nnnnH)*" is displayed and the link process generated no errors or a C program is run and the message appears for FILE "entry". A comp_db file does not exist for the assembly language file. No source data can be returned.

Solution: None.

Problem: An apparently correct variable or procedure name is included in a trace specification and the error message "*Variable not found:<VAR>" or "Module not found:<MODULE>*" is displayed. User error.

Solution: Check all elements of the path given for the variable or procedure. Make sure all elements of the path are there and are correct. Verify that the default path and loaded absolute refer to the same file.

Problem: A "*Bad line range*" error message is displayed at execution of a trace measurement. The lines specified may be at the same address in the program.

Solution: Increment the last line in the range.

Problem: An apparently correct command does not work when a lower case identifier is used. The lower case identifier is identical to one of the software analyzer commands. Identifiers cannot have the same name as software analyzer commands.

Solution: Put the lower case identifier in quotes ("), e.g. "entry".

Problem: The error message "*Program execution outside of absolute file (PC=nnnnH)*" is displayed and no symbolic information is shown. The software analyzer does not display symbolic information when more than one absolute file is loaded and files in any absolute file other than the last one loaded are being traced. The software analyzer can provide symbolic information only for files in the last absolute file loaded.

Solution: Ensure that the files you want to trace are in the absolute file listed on the software analyzer setup display.

Problem: The error message "*Symbol not found*" is displayed for a known C variable. The specified variable is a block variable (a variable defined within an inner block of a procedure). The software analyzer does not support block variables.

Solution: Declare the variable at the procedure level.

Unexpected Source Line

Problem: The source line is not the line associated with the symbol in a trace variables measurement. The software analyzer shows a prefetched line as causing the access.

Solution: The correct source line is the previous source line displayed in the trace.

Problem: Comments are displayed as source lines. The comment spans multiple lines.

Solution: Start and end each comment line with a comment delimiter.

Problem: The source statement shown at the beginning of a procedure is an "end" statement from the previous procedure. The compiler overhead for entry into the procedure is executing. The source line is undefined for these instructions, however, the line preceding the procedure declaration is shown.

Solution: None

Problem: A state position is requested and an unexpected state position is displayed.

Solution: Use a position from the status line rather than a randomly selected state position.

Problem: A prefetched source line immediately following a looping construct is displayed in the looping construct in a trace statements measurement.

Solution: None

Unexpected Symbols On The Display

Problem: A variable name is shown on the display that is different than the expected name. When a variable is passed by reference to a procedure, the procedure accesses that variable under an alias.

Solution: None.

Problem: Extra accesses to a record are shown without a field name or with an extra index to an array field. This is caused by pad bytes the compiler inserts in records to align the fields.

Solution: Create records that don't have pad bytes.

Problem: Unexpected reads/writes at the beginning or end of a procedure caused by compiler overhead.

Solution: Check an assembly listing file of the program. These reads/writes may be operations done by the compiler to set up stacks, transfer parameters, etc.

Unexpected Value On The Display

Problem: The value of the symbol is illegal for the symbol type.

Solution: None.

Problem: Incomplete access to variables. This can occur when the software analyzer is tracing two 32-bit integers and one is assigned to the other or when the software analyzer is tracing two structured variables where one is assigned to the other.

Solution: The complete value can be determined by noting the partial values and their locations. Alternatively, set up a break on the next source line after the assignment and display the variable.

Problem: Partial values are displayed or incorrect complete values in the trace variables measurement. The trace data begins part of the way through a variable access or the user positions the display to the middle of a variable access.

Solution: Use the display positioning to find a location that makes the partial values disappear. If that doesn't work, piece together the whole value from the partial values. Alternatively, setup a break on the next source line after the assignment and display the variable.

Problem: The absolute count does not change between lines although source lines are being shown. Resolution of the count causes small changes to disappear.

Solution: Display the count field in relative mode to see the count between lines.

INDEX

a

b

c

d

e

f

g

h

i

l

m

n

o

p

r

S

t

u

v

w

FOLD HERE

HEWLETT PACKARD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD

Logic Product Support Dept.
Attn: Technical Publications Manager
Centennial Annex - D2
P.O. Box 617
Colorado Springs, Colorado 80901-0617

FOLD HERE

Your cooperation in completing and returning this form will be greatly appreciated. Thank you.

READER COMMENT SHEET

Operating Manual, Model 64341
Real-Time High Level Software Analyzer 68000/68010
64341-90903 E0985, September 1985

Your comments are important to us. Please answer this questionnaire and return it to us. Circle the number that best describes your answer in questions 1 through 7. Thank you.

1. The information in this book is complete:

Doesn't cover enough (what more do you need?)	1 2 3 4 5	Covers everything

2. The information in this book is accurate:

Too many errors	1 2 3 4 5	Exactly right

3. The information in this book is:

Difficult to find	1 2 3 4 5	Easy to find

4. The Index and Table of Contents are useful:

Missing or inadequate	1 2 3 4 5	Helpful

5. What about the "how-to" procedures and examples:

No help	1 2 3 4 5	Very Helpful
Not enough	1 2 3 4 5	Too many

6. What about the writing style:

Confusing	1 2 3 4 5	Clear

7. What about organization of the book:

Poor order	1 2 3 4 5	Good order

8. What about the size of the book:

Too small	1 2 3 4 5	Too big

Comments: _____
_____
_____
_____
_____
_____
Particular pages with errors? _____
Name: _____
Job title: _____
Company: _____
Address: _____

Note: If mailed outside U.S.A., place card in envelope. Use address shown on other side of this card.

FOLD HERE

HEWLETT PACKARD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 1303 COLORADO SPRINGS, COLORADO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD

Logic Product Support Dept.
Attn: Technical Publications Manager
Centennial Annex - D2
P.O. Box 617
Colorado Springs, Colorado 80901-0617

FOLD HERE

Your cooperation in completing and returning this form will be greatly appreciated. Thank you.

READER COMMENT SHEET

Operating Manual, Model 64341
Real-Time High Level Software Analyzer 68000/68010
64341-90903 E0985, September 1985

Your comments are important to us. Please answer this questionnaire and return it to us. Circle the number that best describes your answer in questions 1 through 7. Thank you.

1. The information in this book is complete:

Doesn't cover enough (what more do you need?)	1 2 3 4 5	Covers everything

2. The information in this book is accurate:

Too many errors	1 2 3 4 5	Exactly right

3. The information in this book is:

Difficult to find	1 2 3 4 5	Easy to find

4. The Index and Table of Contents are useful:

Missing or inadequate	1 2 3 4 5	Helpful

5. What about the "how-to" procedures and examples:

No help	1 2 3 4 5	Very Helpful
Not enough	1 2 3 4 5	Too many

6. What about the writing style:

Confusing	1 2 3 4 5	Clear

7. What about organization of the book:

Poor order	1 2 3 4 5	Good order

8. What about the size of the book:

Too small	1 2 3 4 5	Too big

Comments: _____
_____
_____
_____
_____
_____
Particular pages with errors? _____
Name: _____
Job title: _____
Company: _____
Address: _____

Note: If mailed outside U.S.A., place card in envelope. Use address shown on other side of this card.

SALES & SUPPORT OFFICES
Arranged alphabetically by country

Product Line Sales/Support Key

Key Product Line
A Analytical
CM Components
C Computer Systems
E Electronic Instruments & Measurement Systems
M Medical Products
P Personal Computation Products
*** Sales only for specific product line**
**** Support only for specific product line**

IMPORTANT:These symbols designate general product line capability.They do not insure sales or support availability for all products within a line, at all locations.Contact your local sales office for information regarding locations where HP support is available for specific products.

HEADQUARTERS OFFICES

If there is no sales office listed for your area, contact one of these headquarters offices.

NORTH/CENTRAL AFRICA
Hewlett-Packard S.A.
7, rue du Bois-du-Lan
CH-1217 **MEYRIN** 1, Switzerland
Tel: (022) 83 12 12
Telex: 27835 hmea
Cable: HEWPACKSA Geneve

ASIA
Hewlett-Packard Asia Ltd.
47/F, 26 Harbour Rd.,
Wanchai, **HONG KONG**
G.P.O. Box 863, Hong Kong
Tel: 5-8330833
Telex: 76793 HPA HX
Cable: HPASIAL TD

CANADA
Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
Telex: 610-492-4246

EASTERN EUROPE
Hewlett-Packard Ges.m.b.h.
Lieblgasse 1
P.O.Box 72
A-1222 **VIENNA,** Austria
Tel: (222) 2500-0
Telex: 1 3 4425 HEPA A

NORTHERN EUROPE
Hewlett-Packard S.A.
Uilenstede 475
P.O.Box 999
NL-1183 AG **AMSTELVEEN**
The Netherlands
Tel: 20 437771
Telex: 18 919 hpner nl

SOUTH EAST EUROPE
Hewlett-Packard S.A.
World Trade Center
110 Avenue Louis Casai
1215 Cointrin, **GENEVA,** Switzerland
Tel: (022) 98 96 51
Telex: 27225 hpser

MEDITERRANEAN AND MIDDLE EAST
Hewlett-Packard S.A.
Mediterranean and Middle East Operations
Atrina Centre
32 Kifissias Ave.
Paradissos-Amarousion, **ATHENS**
Greece
Tel: 682 88 11
Telex: 21-6588 HPAT GR
Cable: HEWPACKSA Athens

UNITED KINGDOM
Hewlett-Packard Ltd.
Nine Mile Ride
Easthampstead, **WOKINGHAM**
Berkshire, IRGll 3LL
Tel: 0344 773100
Telex: 848805

EASTERN USA
Hewlett-Packard Co.
4 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 258-2000

MIDWESTERN USA
Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800

SOUTHERN USA
Hewlett-Packard Co.
2000 South Park Place
P.O. Box 105005
ATLANTA, GA 30348
Tel: (404) 955-1500

WESTERN USA
Hewlett-Packard Co.
3939 Lankershim Blvd.
P.O. Box 3919
LOS ANGELES, CA 91604
Tel: (213) 506-3700

OTHER INTERNATIONAL AREAS
Hewlett-Packard Co.
Intercontinental Headquarters
3495 Deer Creek Road
PALO ALTO, CA 94304
Tel: (415) 857-1501
Telex: 034-8300
Cable: HEWPACK

ANGOLA
Telectra Angola LDA
Empresa Tecnica de Equipamentos
Rua Conselheiro Julio de Vilhema, 16
Caixa Postal 6487
LUANDA
Tel: 35515,35516
Telex: 3134
E,C*

ARGENTINA
Hewlett-Packard Argentina S.A.
Montaneses 2140/50
1428 **BUENOS AIRES**
Tel: 783-4886/4836/4730
Cable: HEWPACKARG
A,C,CM,E,P

Biotron S.A.C.I.e.I.
Av. Paso Colon 221, Piso 9
1399 **BUENOS AIRES**
CM

Laboratorio Rodriguez
Corswant S.R.L.
Misiones, 1156 - 1876
Bernal, Oeste
BUENOS AIRES
Tel: 252-3958, 252-4991
A

Argentina Esanco S.R.L.
Avasco 2328
1416 **BUENOS AIRES**
Tel: 541-58-1981, 541-59-2767
A

AUSTRALIA

Adelaide, South Australia Office
Hewlett-Packard Australia Ltd.
153 Greenhill Road
PARKSIDE, S.A. 5063
Tel: 272-5911
Telex: 82536
Cable: HEWPARD Adelaide
A*,C,CM,E,M,P

Brisbane, Queensland Office
Hewlett-Packard Australia Ltd.
10 Payne Road
THE GAP, Queensland 4061
Tel: 30-4133
Telex: 42133
Cable: HEWPARD Brisbane
A,C,CM,E,M,P

Canberra, Australia Capital Territory Office
Hewlett-Packard Australia Ltd.
121 Wollongong Street
FYSHWICK, A.C.T. 2609
Tel: 80 4244
Telex: 62650
Cable: HEWPARD Canberra
C,CM,E,P

Melbourne, Victoria Office
Hewlett-Packard Australia Ltd.
31-41 Joseph Street
BLACKBURN, Victoria 3130
Tel: 895-2895
Telex: 31-024
Cable: HEWPARD Melbourne
A,C,CM,E,M,P

Perth, Western Australia Office
Hewlett-Packard Australia Ltd.
261 Stirling Highway
CLAREMONT, W.A. 6010
Tel: 383-2188
Telex: 93859
Cable: HEWPARD Perth
A,C,CM,E,M,P

Sydney, New South Wales Office
Hewlett-Packard Australia Ltd.
17-23 Talavera Road
P.O. Box 308
NORTH RYDE, N.S.W. 2113
Tel: 888-4444
Telex: 21561
Cable: HEWPARD Sydney
A,C,CM,E,M,P

AUSTRIA
Hewlett-Packard Ges.m.b.h.
Verkauisbüro Graz
Grottenhofstrasse 94
A-8052 **GRAZ**
Tel: (0316) 291 56 60
Telex: 32375
C,E

Hewlett-Packard Ges.m.b.h.
Lieblgasse 1
P.O. Box 72
A-1222 **VIENNA**
Tel: (0222) 2500-0
Telex: 134425 HEPA A
A,C,CM,E,M,P

BAHRAIN
Green Salon
P.O. Box 557
MANAMA
Tel: 255503-255950
Telex: 8441
P

Wael Pharmacy
P.O. Box 648
MANAMA
Tel: 256123
Telex: 8550 WAEL BN
E,M

Zayani Computer Systems
218 Shaik Mubarak Building
Government Avenue
P.O. Box 5918
MANAMA
Tel: 276278
Telex: 9015
P

BELGIUM
Hewlett-Packard Belgium S.A./N.V.
Blvd de la Woluwe, 100
Woluwedal
B-1200 **BRUSSELS**
Tel: (02) 762-32-00
Telex: 23-494 paloben bru
A,C,CM,E,M,P

BERMUDA
Applied Computer Technologies
Atlantic House Building
Par-La-Ville Road
Hamilton 5
Tel: 295-1616
P

BRAZIL
Hewlett-Packard do Brasil
I.e.C. Ltda.
Alameda Rio Negro, 750
Alphaville
06400 **BARUERI SP**
Tel: (011) 421.1311
Telex: (011) 33872 HPBR-BR
Cable: HEWPACK Sao Paulo
A,C,CM,E,M,P

Hewlett-Packard do Brasil
I.e.C. Ltda.
Praia de Botafago 228
6° Andar-conj 614
Edificio Argentina - Ala A
22250 **RIO DE JANEIRO**
Tel: (02l) 552-6422
Telex: 21905 HPBR-BR
Cable: HEWPACK Rio de Janeiro
A,C,CM,E,P*

Convex/Van Den
Rua Jose Bonifacio
458 Todos Os Santos
CEP 20771
RIO DE JANEIRO, RJ
Tel: 591-0197
Telex: 33487 EGLB BR
A

ANAMED I.C.E.I. Ltda.
Rua Bage, 103
04012 **SAO PAULO,** SP
Tel: (011) 572-6537
Telex: 24720 HPBR-BR
M

Datatronix Electronica Ltda.
Av. Pacaembu 746-C11
SAO PAULO, SP
Tel: (118) 260111
CM

CAMEROON
Beriac
B. P. 23
DOUALA
Tel: 420153
Telex: 5351
C,P

CANADA

Alberta
Hewlett-Packard (Canada) Ltd.
3030 3rd Avenue N.E.
CALGARY, Alberta T2A 6T7
Tel: (403) 235-3100
A,C,CM,E*,M,P*

Hewlett-Packard (Canada) Ltd.
11120-178th Street
EDMONTON, Alberta T5S 1P2
Tel: (403) 486-6666
A,C,CM,E,M,P

SALES & SUPPORT OFFICES
Arranged alphabetically by country

CANADA (Cont'd)

British Columbia
Hewlett-Packard (Canada) Ltd.
10691 Shellbridge Way
RICHMOND,
British Columbia V6X 2W7
Tel: (604) 270-2277
Telex: 610-922-5059
A,C,CM,E*,M,P*

Hewlett-Packard (Canada) Ltd.
121 - 3350 Douglas Street
VICTORIA, British Columbia V8Z 3L1
Tel: (604) 381-6616
C

Manitoba
Hewlett-Packard (Canada) Ltd.
1825 Inkster Blvd.
WINNIPEG, Manitoba R2X 1R3
Tel: (204) 694-2777
A,C,CM,E,M,P*

New Brunswick
Hewlett-Packard (Canada) Ltd.
814 Main Street
MONCTON, New Brunswick E1C 1E6
Tel: (506) 855-2841
C

Nova Scotia
Hewlett-Packard (Canada) Ltd.
Suite 111
900 Windmill Road
DARTMOUTH, Nova Scotia B3B 1P7
Tel: (902) 469-7820
C,CM,E*,M,P*

Ontario
Hewlett-Packard (Canada) Ltd.
3325 N. Service Rd., Unit 3
BURLINGTON, Ontario L7N 3G2
Tel: (416) 335-8644
C,M*

Hewlett-Packard (Canada) Ltd.
496 Days Road
KINGSTON, Ontario K7M 5R4
Tel: (613) 384-2088
C

Hewlett-Packard (Canada) Ltd.
552 Newbold Street
LONDON, Ontario N6E 2S5
Tel: (519) 686-9181
A,C,CM,E*,M,P*

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
MISSISSAUGA, Ontario L4V 1M8
Tel: (416) 678-9430
A,C,CM,E,M,P

Hewlett-Packard (Canada) Ltd.
2670 Queensview Dr.
OTTAWA, Ontario K2B 8K1
Tel: (613) 820-6483
A,C,CM,E*,M,P*

Hewlett-Packard (Canada) Ltd.
The Oaks Plaza, Unit #9
2140 Regent Street
SUDBURY, Ontario, P3E 5S8
Tel: (705) 522-0202
C

Hewlett-Packard (Canada) Ltd.
3790 Victoria Park Ave.
WILLOWDALE, Ontario M2H 3H7
Tel: (416) 499-2550
C

Quebec
Hewlett-Packard (Canada) Ltd.
17500 Trans Canada Highway
South Service Road
KIRKLAND, Quebec H9J 2X8
Tel: (514) 697-4232
A,C,CM,E,M,P*

Hewlett-Packard (Canada) Ltd.
1150 rue Claire Fontaine
QUEBEC CITY, Quebec G1R 5G4
Tel: (418) 648-0726
C

Hewlett-Packard (Canada) Ltd.
130 Robin Crescent
SASKATOON, Saskatchewan S7L 6M7
Tel: (306) 242-3702
C

CHILE
ASC Ltda.
Austria 2041
SANTIAGO
Tel: 223-5946, 223-6148
Telex: 340192 ASC CK
C,P

Isical Ltda.
Av. Italia 634 Santiago
Casilla 16475
SANTIAGO 9
Tel: 222-0222
Telex: 440283 JCYCL CZ
CM,E,M

Metrolab S.A.
Monjitas 454 of. 206
SANTIAGO
Tel: 395752, 398296
Telex: 340866 METLAB CK
A

Olympia (Chile) Ltda.
Av. Rodrigo de Araya 1045
Casilla 256-V
SANTIAGO 21
Tel: 225-5044
Telex: 340892 OLYMP
Cable: Olympiachile Santiagochile
C,P

CHINA, People's Republic of
China Hewlett-Packard, Ltd.
47/F China Resources Bldg.
26 Harbour Road
HONG KONG
Tel: 5-8330833
Telex: 76793 HPA HX
Cable: HP ASIA LTD
A*,M*

China Hewlett-Packard, Ltd.
P.O. Box 9610, Beijing
4th Floor, 2nd Watch Factory Main Bldg.
Shuang Yu Shu, Bei San Huan Rd.
Hai Dian District
BEIJING
Tel: 28-0567
Telex: 22601 CTSHP CN
Cable: 1920 Beijing
A,C,CM,E,M,P

COLOMBIA
Instrumentación
H. A. Langebaek & Kier S.A.
Carrera 4A No. 52A-26
Apartado Aereo 6287
BOGOTA 1, D.E.
Tel: 212-1466
Telex: 44400 INST CO
Cable: AARIS Bogota
CM,E,M

Nefromedicas Ltda.
Calle 123 No. 9B-31
Apartado Aereo 100-958
BOGOTA D.E., 10
Tel: 213-5267, 213-1615
Telex: 43415 HEGAS CO
A

Compumundo
Avenida 15 # 107-80
BOGOTA D.E.
Tel: 214-4458
Telex: 45466 MARICO
P

Carvajal, S.A.
Calle 29 Norte No. 6A-40
Apartado Aereo 46
CALI
Tel: 368-1111
Telex: 55650
C,E,P

CONGO
Seric-Congo
B. P. 2105
BRAZZAVILLE
Tel: 815034
Telex: 5262

COSTA RICA
Cientifica Costarricense S.A.
Avenida 2, Calle 5
San Pedro de Montes de Oca
Apartado 10159
SAN JOSÉ
Tel: 24-38-20, 24-08-19
Telex: 2367 GALGUR CR
CM,E,M

CYPRUS
Telerexa Ltd.
P.O. Box 4809
14C Stassinos Avenue
NICOSIA
Tel: 62698
Telex: 2894 LEVIDO CY
E,M,P

DENMARK
Hewlett-Packard A/S
Datavej 52
DK-3460 **BIRKEROD**
Tel: (02) 81-66-40
Telex: 37409 hpas dk
A,C,CM,E,M,P

Hewlett-Packard A/S
Rolighedsvej 32
DK-8240 **RISSKOV,** Aarhus
Tel: (06) 17-60-00
Telex: 37409 hpas dk
C,E

DOMINICAN REPUBLIC
Microprog S.A.
Juan Tomás Mejía y Cotes No. 60
Arroyo Hondo
SANTO DOMINGO
Tel: 565-6268
Telex: 4510 ARENTA DR (RCA)
P

ECUADOR
CYEDE Cia. Ltda.
Avenida Eloy Alfaro 1749
y Belgica
Casilla 6423 CCI
QUITO
Tel: 450-975, 243-052
Telex: 22548 CYEDE ED
CM,E,P

Medtronics
Valladolid 524 Madrid
P.O. 9171, **QUITO**
Tel: 223-8951
Telex: 2298 ECKAME ED
A

Hospitalar S.A.
Robles 625
Casilla 3590
QUITO
Tel: 545-250, 545-122
Telex: 2485 HOSPTL ED
Cable: HOSPITALAR-Quito
M

Ecuador Overseas Agencies C.A.
Calle 9 de Octubre #818
P.O. Box 1296, Guayaquil
QUITO
Tel: 306022
Telex: 3361 PBCGYE ED
M

EGYPT
Sakrco Enterprises
70, Mossadak Str.
Dokki, Giza
CAIRO
Tel: 706440
Telex: 93146
C

International Engineering Associates
24 Hussein Hegazi Street
Kasr-el-Ain
CAIRO
Tel: 23829, 21641
Telex: 93830 IEA UN
Cable: INTEGASSO
E,M*

S.S.C. Medical
40 Gezerat El Arab Street
Mohandessin
CAIRO
Tel: 803844, 805998, 810263
Telex: 20503 SSC UN
M*

EL SALVADOR
IPESA de El Salvador S.A.
29 Avenida Norte 1223
SAN SALVADOR
Tel: 26-6858, 26-6868
Telex: 20539 IPESA SAL
A,C,CM,E,P

ETHIOPIA
Seric-Ethiopia
P.O. Box 2764
ADDIS ABABA
Tel: 185114
Telex: 21150
C,P

FINLAND
Hewlett-Packard Oy
Piispankalliontie 17
02200 **ESPOO**
Tel: 00358-0-88721
Telex: 121563 HEWPA SF
A,C,CM,E,M,P

FRANCE
Hewlett-Packard France
Z.I. Mercure B
Rue Berthelot
13763 Les Milles Cedex
AIX-EN-PROVENCE
Tel: (42) 59-41-02
Telex: 410770F
A,C,E,M,P*

Hewlett-Packard France
64, rue Marchand Saillant
61000 **ALENCON**
Tel: (33) 29 04 42

Hewlett-Packard France
28 rue de la Republique
Boite Postale 503
25026 **BESANCON** Cedex
Tel: (81) 83-16-22
Telex: 361157
C,M

Hewlett-Packard France
Chemin des Mouilles
Boite Postale 162
69130 **ECULLY** Cedex (Lyon)
Tel: (78) 833-81-25
Telex: 310617F
A,C,E,M

Hewlett-Packard France
Parc d'activités du Bois Briard
2, avenue du Lac
91040 **EVRY** Cedex
Tel: 6 077-96 60
Telex: 692315F
E

Hewlett-Packard France
5, avenue Raymond Chanas
38320 **EYBENS** (Grenoble)
Tel: (76) 62-57-98
Telex: 980124 HP GRENOB EYBE
C

Hewlett-Packard France
Rue Fernand. Forest
Z.A. Kergaradec
29239 **GOUESNOU**
Tel: (98) 41-87-90

Hewlett-Packard France
Centre d'affaires Paris-Nord
Bâtiment Ampère
Rue de la Commune de Paris
Boite Postale 300
93153 **LE BLANC-MESNIL**
Tel: (1) 865-44-52
Telex: 211032F
C,E,M

Hewlett-Packard France
Parc d'activités Cadera
Quartier Jean-Mermoz
Avenue du Président JF Kennedy
F-33700 **MÉRIGNAC** (Bordeaux)
Tel: (56) 34-00-84
Telex: 550105F
C,E,M

Hewlett-Packard France
Immueble "Les 3 B"
Nouveau chemin de la Garde
ZAC du Bois Briand
44085 **NANTES** Cedex
Tel: (40) 50-32-22
Telex: 711085F
C**

Hewlett-Packard France
125, rue du Faubourg Bannier
45000 **ORLEANS**
Tel: (38) 68 01 63

FRANCE (Cont'd)

Hewlett-Packard France
Zone Industrielle de Courtaboeuf
Avenue des Tropiques
91947 Les Ulis Cedex **ORSAY**
Tel: (6) 907-78-25
Telex: 600048F
A,C,CM,E,M,P

Hewlett-Packard France
Paris Porte-Maillot
15, boulevard de L'Amiral-Bruix
75782 **PARIS** Cedex 16
Tel: (1) 502-12-20
Telex: 613663F
C,M,P

Hewlett-Packard France
124, Boulevard Tourasse
64000 **PAU**
Tel: (59) 80 38 02

Hewlett-Packard France
2 Allée de la Bourgonnette
35100 **RENNES**
Tel: (99) 51-42-44
Telex: 740912F
C,CM,E,M,P*

Hewlett-Packard France
98 avenue de Bretagne
76100 **ROUEN**
Tel: (35) 63-57-66
Telex: 770035F
C

Hewlett-Packard France
4, rue Thomas-Mann
Boite Postale 56
67033 **STRASBOURG** Cedex
Tel: (88) 28-56-46
Telex: 890141F
C,E,M,P*

Hewlett-Packard France
La Péripole III
20, chemin du Pigeonnier de la Cépière
F-31083 **TOULOUSE** Cedex
Tel: (61) 40-11-12
Telex: 531639F
A,C,E,P*

Hewlett-Packard France
9, rue Baudin
26000 **VALENCE**
Tel: (75) 42 76 16

Hewlett-Packard France
Carolor
ZAC de Bois Briand
57640 **VIGY** (Metz)
Tel: (8) 771 20 22
C

Hewlett-Packard France
Parc d'activité des Prés
1, rue Papin
59658 **VILLENEUVE D'ASCQ** Cedex
Tel: (20) 47 78 78
Telex: 160124F
C,E,M,P*

GABON

Sho Gabon
P.O. Box 89
LIBREVILLE
Tel: 721 484
Telex: 5230

GERMAN FEDERAL REPUBLIC

Hewlett-Packard GmbH
Geschäftsstelle
Keithstrasse 2-4
D-1000 **BERLIN** 30
Tel: (030) 21 99 04-0
Telex: 018 3405 hpbln d
A,C,E,M,P

Hewlett-Packard GmbH
Vertriebszentrun Südwest
Schickardstrasse 2
D-7030 **BÖBLINGEN**
Tel: (07031) 645-0
Telex: 7265 743 hep
A,C,CM,E,M,P

Hewlett-Packard GmbH
Vertriebszentrum West
Berliner Strasse III
D-4030 **RATINGEN** 3
Tel: (02102) 494-0
Telex: 589 070 hprad
A,C,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Schleefstr. 28a
D-4600 **DORTMUND**-41
Tel: (0231) 45001
Telex: 822858 hepdad
A,C,E

Hewlett-Packard GmbH
Vertriebszentrum Mitte
Hewlett-Packard-Strasse
D-6380 **BAD HOMBURG**
Tel: (06172) 400-0
Telex: 410 844 hpbhg
A,C,E,M,P

Hewlett-Packard GmbH
Vertriebszentrum Nord
Kapstadtring 5
D-2000 **HAMBURG** 60
Tel: (040) 63804-1
Telex: 021 63 032 hphh d
A,C,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Heidering 37-39
D-3000 **HANNOVER** 61
Tel: (0511) 5706-0
Telex: 092 3259
A,C,CM,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Rosslauer Weg 2-4
D-6800 **MANNHEIM**
Tel: (0621) 70 05-0
Telex: 0462105
A,C,E

Hewlett-Packard GmbH
Geschäftsstelle
Messerschmittstrasse 7
D-7910 **NEU ULM**
Tel: (0731) 70 73-0
Telex: 0712816 HP ULM-D
A,C,E*

Hewlett-Packard GmbH
Geschäftsstelle
Emmericher Strasse 13
D-8500 **NÜRNBERG** 10
Tel: (0911) 5205-0
Telex: 0623 860 hpnbg
C,CM,E,M,P

Hewlett-Packard GmbH
Vertriebszentrum Süd
Eschenstrasse 5
D-8028 **TAUFKIRCHEN**
Tel: (089) 61 20 7-0
Telex: 0524985
A,C,CM,E,M,P

Hewlett-Packard GmbH
Geschäftsstelle
Ermlisallee
7517 **WALDBRONN** 2
Tel: (07243) 602-0
Telex: 782 838 hepk
A,C,E

GREAT BRITAIN

See United Kingdom

GREECE

Hewlett-Packard A.E.
178, Kifissias Avenue
6th Floor
Halandri-**ATHENS**
Greece
Tel: 6471543, 6471673, 6472971
Telex: 221 286 HPHLGR
A,C,CM**,E,M,P

Kostas Karaynnis S.A.
8, Omirou Street
ATHENS 133
Tel: 32 30 303, 32 37 371
Telex: 215962 RKAR GR
A,C*,CM,E

Impexin
Intelect Div.
209 Mesogion
11525 **ATHENS**
Tel: 6474481/2
Telex: 216286
P

Haril Company
38, Mihalakopoulou
ATHENS 612
Tel: 7236071
Telex: 218767
M*

Hellamco
P.O. Box 87528
18507 **PIRAEUS**
Tel: 4827049
Telex: 241441
A

GUATEMALA

IPESA
Avenida Reforma 3-48, Zona 9
GUATEMALA CITY
Tel: 316627, 314786
Telex: 3055765 IPESA GU
A,C,CM,E,M,P

HONG KONG

Hewlett-Packard Hong Kong, Ltd.
G.P.O. Box 795
5th Floor, Sun Hung Kai Centre
30 Harbour Road
HONG KONG
Tel: 5-8323211
Telex: 66678 HEWPA HX
Cable: HEWPACK Hong Kong
E,C,P

CET Ltd.
10th Floor, Hua Asia Bldg.
64-66 Gloulester Road
HONG KONG
Tel: (5) 200922
Telex: 85148 CET HX
CM

Schmidt & Co. (Hong Kong) Ltd.
18th Floor, Great Eagle Centre
23 Harbour Road
HONG KONG
Tel: 5-8330222
Telex: 74766 SCHMC HX
A,M

ICELAND

Hewlett-Packard Iceland
Hoefdabakka 9
110 **Reykjavik**
Tel: (1) 67 1000
A,C,CM,E,M,P

INDIA

Computer products are sold through Blue Star Ltd. All computer repairs and maintenance service is done through Computer Maintenance Corp.

Blue Star Ltd.
Sabri Complex 2nd Floor
24 Residency Rd.
BANGALORE 560 025
Tel: 55660, 578881
Telex: 0845-430
Cable: BLUESTAR
A,C*,CM,E

Blue Star Ltd.
Band Box House
Prabhadevi
BOMBAY 400 025
Tel: 4933101, 4933222
Telex: 011-71051
Cable: BLUESTAR
A,M

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
BOMBAY 400 025
Tel: 422-6155, 422-6556
Telex: 011-71193 BSSS IN
Cable: FROSTBLUE
A,C*,CM,E,M

Blue Star Ltd.
Kalyan, 19 Vishwas Colony
Alkapuri, **BORODA,** 390 005
Tel: 65235, 65236
Cable: BLUE STAR
A

Blue Star Ltd.
7 Hare Street
CALCUTTA 700 001
Tel: 230131, 230132
Telex: 021-7655
Cable: BLUESTAR
A,M

Blue Star Ltd.
133 Kodambakkam High Road
MADRAS 600 034
Tel: 472056, 470238
Telex: 041-379
Cable: BLUESTAR
A,M

Blue Star Ltd.
13 Community Center
New Friends Colony
NEW DELHI 110 065
Tel: 633773, 634473
Telex: 031-61120
Cable: BLUEFROST
A,C*,CM,E,M

Blue Star Ltd.
15/16 C Wellesley Rd.
PUNE 411 011
Tel: 22775
Cable: BLUE STAR
A

Blue Star Ltd.
2-2-47/1108 Bolarum Rd.
SECUNDERABAD 500 003
Tel: 72057, 72058
Telex: 0155645
Cable: BLUEFROST
A,E

Blue Star Ltd.
T.C. 7/603 Poornima
Maruthunkuzhi
TRIVANDRUM 695 013
Tel: 65799, 65820
Telex: 0884-259
Cable: BLUESTAR
E

Computer Maintenance Corporation Ltd.
115, Sarojini Devi Road
SECUNDERABAD 500 003
Tel: 310-184, 345-774
Telex: 031-2960
C**

INDONESIA

BERCA Indonesia P.T.
P.O.Box 496/Jkt.
Jl. Abdul Muis 62
JAKARTA
Tel: 21-373009
Telex: 46748 BERSAL IA
Cable: BERSAL JAKARTA
P

BERCA Indonesia P.T.
P.O.Box 2497/Jkt
Antara Bldg., 12th Floor
Jl. Medan Merdeka Selatan 17
JAKARTA-PUSAT
Tel: 21-340417, 341445
Telex: 46748 BERSAL IA
A,C,E,M

BERCA Indonesia P.T.
Jalan Kutai 24
SURABAYA
Tel: 67118
Telex: 31146 BERSAL SB
Cable: BERSAL-SURABAYA
A*,E,M,P

IRAQ

Hewlett-Packard Trading S.A.
Service Operation
Al Mansoor City 9B/3/7
BAGHDAD
Tel: 551-49-73
Telex: 212-455 HEPAIRAQ IK
C

IRELAND

Hewlett-Packard Ireland Ltd.
82/83 Lower Leeson Street
DUBLIN 2
Tel: 0001 608800
Telex: 30439
A,C,CM,E,M,P

Cardiac Services Ltd.
Kilmore Road
Artane
DUBLIN 5
Tel: (01) 351820
Telex: 30439
M

SALES & SUPPORT OFFICES
Arranged alphabetically by country

ISRAEL
Eldan Electronic Instrument Ltd.
P.O.Box 1270
JERUSALEM 91000
16, Ohaliav St.
JERUSALEM 94467
Tel: 533 221, 553 242
Telex: 25231 AB/PAKRD IL
A,M

Computation and Measurement Systems (CMS) Ltd.
11 Masad Street
67060
TEL-AVIV
Tel: 388 388
Telex: 33569 Motil IL
C,CM,E,P

ITALY
Hewlett-Packard Italiana S.p.A
Traversa 99C
Via Giulio Petroni, 19
I-70124 **BARI**
Tel: (080) 41-07-44
C,M

Hewlett-Packard Italiana S.p.A.
Via Emilia, 51/C
I-40011 **BOLOGNA** Anzola Dell'Emilia
Tel: (051) 731061
Telex: 511630
C,E,M

Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43G/C
I-95126 **CATANIA**
Tel: (095) 37-10-87
Telex: 970291
C

Hewlett-Packard Italiana S.p.A.
Via G. Di Vittorio 9
I-20063 **CERNUSCO SUL NAVIGLIO**
(Milano)
Tel: (02) 4459041
Telex: 334632
A,C,CM,E,M,P

Hewlett-Packard Italiana S.p.A.
Via C. Colombo 49
I-20090 **TREZZANO SUL NAVIGLIO**
(Milano)
Tel: (02) 4459041
Telex: 322116
C

Hewlett-Packard Italiana S.p.A.
Via Nuova San Rocco a Capodimonte, 62/A
I-80131 **NAPOLI**
Tel: (081) 7413544
Telex: 710698
A**,C,E,M

Hewlett-Packard Italiana S.p.A.
Viale G. Modugno 33
I-16156 **GENOVA PEGLI**
Tel: (010) 68-37-07
Telex: 215238
C,E

Hewlett-Packard Italiana S.p.A.
Via Pelizzo 15
I-35128 **PADOVA**
Tel: (049) 664888
Telex: 430315
A,C,E,M

Hewlett-Packard Italiana S.p.A.
Viale C. Pavese 340
I-00144 **ROMA EUR**
Tel: (06) 54831
Telex: 610514
A,C,E,M,P*

Hewlett-Packard Italiana S.p.A.
Via di Casellina 57/C
I-50018 **SCANDICCI-FIRENZE**
Tel: (055) 753863
C,E,M

Hewlett-Packard Italiana S.p.A.
Corso Svizzera, 185
I-10144 **TORINO**
Tel: (011) 74 4044
Telex: 221079
A*,C,E

IVORY COAST
S.I.T.E.L.
Societe Ivoirienne de Telecommunications
Bd. Giscard d'Estaing
Carrefour Marcory
Zone 4.A.
Boite postale 2580
ABIDJAN 01
Tel: 353600
Telex: 43175
E

S.I.T.I.
Immeuble "Le General"
Av. du General de Gaulle
01 BP 161
ABIDJAN 01
Tel: 321227
C,P

JAPAN
Yokogawa-Hewlett-Packard Ltd.
152-1, Onna
ATSUGI, Kanagawa, 243
Tel: (0462) 25-0031
C,CM,E

Yokogawa-Hewlett-Packard Ltd.
Meiji-Seimei Bldg. 6F
3-1 Hon Chiba-Cho
CHIBA, 280
Tel: 472 25 7701
C,E

Yokogawa-Hewlett-Packard Ltd.
Yasuda-Seimei Hiroshima Bldg.
6-11, Hon-dori, Naka-ku
HIROSHIMA, 730
Tel: 82-241-0611

Yokogawa-Hewlett-Packard Ltd.
Towa Building
2-3, Kaigan-dori, 2 Chome Chuo-ku
KOBE, 650
Tel: (078) 392-4791
C,E

Yokogawa-Hewlett-Packard Ltd.
Kumagaya Asahi 82 Bldg
3-4 Tsukuba
KUMAGAYA, Saitama 360
Tel: (0485) 24-6563
C,CM,E

Yokogawa-Hewlett-Packard Ltd.
Asahi Shinbun Daiichi Seimei Bldg.
4-7, Hanabata-cho
KUMAMOTO, 860
Tel: (096) 354-7311
C,E

Yokogawa-Hewlett-Packard Ltd.
Shin-Kyoto Center Bldg.
614, Higashi-Shiokoji-cho
Karasuma-Nishiiru
Shiokoji-dori, Shimogyo-ku
KYOTO, 600
Tel: 075-343-0921
C,E

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsui Bldg
4-73, Sanno-maru, 1 Chome
MITO, Ibaraki 310
Tel: (0292) 25-7470
C,CM,E

Yokogawa-Hewlett-Packard Ltd.
Meiji-Seimei Kokubun Bldg. 7-8
Kokubun, 1 Chome, Sendai
MIYAGI, 980
Tel: (0222) 25-1011
C,E

Yokogawa-Hewlett-Packard Ltd.
Nagoya Kokusai Center Building
47-1, Nagono, 1 Chome
Nakamura-ku
NAGOYA, 450
Tel: (052) 571-5171
C,CM,E,M

Yokogawa-Hewlett-Packard Ltd.
Saikyoren Building
1-2 Dote-machi, **OHMIYA**
Saitama 330
Tel: (0486) 45-8031

Yokogawa-Hewlett-Packard Ltd.
Chuo Bldg.,
4-20 Nishinakajima, 5 Chome
Yodogawa-ku
OSAKA, 532
Tel: (06) 304-6021
Telex: YHPOSA 523-3624
A,C,CM,E,M,P*

Yokogawa-Hewlett-Packard Ltd.
27-15, Yabe, 1 Chome
SAGAMIHARA Kanagawa, 229
Tel: 0427 59-1311

Yokogawa-Hewlett-Packard Ltd.
Daiichi Seimei Bldg.
7-1, Nishi Shinjuku, 2 Chome
Shinjuku-ku,**TOKYO** 160
Tel: 03-348-4611
C,E

Yokogawa-Hewlett-Packard Ltd.
29-21 Takaido-Higashi, 3 Chome
Suginami-ku **TOKYO** 168
Tel: (03) 331-6111
Telex: 232-2024 YHPTOK
A,C,CM,E,M,P*

Yokogawa Hokushin Electric Corp.
9-32 Nokacho 2 Chome
2 Chome Musashino-shi
TOKYO, 180
Tel: (0422) 54-1111
Telex: 02822-421 YEW MTK J
A

Yokogawa-Hewlett-Packard Ltd.
Meiji-Seimei
Utsunomiya Odori Building
1-5 Odori, 2 Chome
UTSUNOMIYA, Tochigi 320
Tel: (0286) 33-1153
C,E

Yokogawa-Hewlett-Packard Ltd.
Yasuda Seimei Yokohama Nishiguchi Bldg.
30-4 Tsuruya-cho, 3 Chome
YOKOHAMA 221
Tel: (045) 312-1252
C,E

JORDAN
Scientific and Medical Supplies Co.
P.O. Box 1387
AMMAN
Tel: 24907, 39907
Telex: 21456 SABCO JO
C,E,M,P

KENYA
ADCOM Ltd., Inc., Kenya
P.O.Box 30070
NAIROBI
Tel: 331955
Telex: 22639
E,M

KOREA
Samsung Hewlett-Packard Co. Ltd.
Dongbang Yeoeuido Building
12-16th Floors
36-1 Yeoeuido-dong
Yongdeungpo-ku
SEOUL
Tel: 784-2666, 784-4666
Telex: 25166 SAMSAN K
A,C,CM,E,M,P

Young In Scientific Co., Ltd.
Youngwha Building
547 Shinsa Dong, Kangnam-ku
SEOUL 135
Tel: 5467771
Telex: K23457 GINSCO
A

KUWAIT
Al-Khaldiya Trading & Contracting
P.O. Box 830
SAFAT
Tel: 424910, 411726
Telex: 22481 AREEG KT
Cable: VISCOUNT
E,M,A

Gulf Computing Systems
P.O. Box 25125
SAFAT
Tel: 435969
Telex: 23648
P

Photo & Cine Equipment
P.O. Box 270
SAFAT
Tel: 2445111
Telex: 22247 MATIN KT
Cable: MATIN KUWAIT
P

W.J. Towell Computer Services
P.O. Box 5897
SAFAT
Tel: 2462640
Telex: 30336 TOWELL KT
C

LEBANON
Computer Information Systems S.A.L.
Chammas Building
P.O. Box 11-6274 Dora
BEIRUT
Tel: 89 40 73
Telex: 42309
C,E,M,P

LIBERIA
Unichemicals Inc.
P.O. Box 4509
MONROVIA
Tel: 224282
Telex: 4509
E

MADAGASCAR
Technique et Precision
12, rue de Nice
P.O. Box 1227
101 **ANTANANARIVO**
Tel: 22090
Telex: 22255
P

LUXEMBOURG
Hewlett-Packard Belgium S.A./N.V.
Blvd de la Woluwe, 100
Woluwedal
B-1200 **BRUSSELS**
Tel: (02) 762-32-00
Telex: 23-494 paloben bru
A,C,CM,E,M,P

MALAYSIA
Hewlett-Packard Sales (Malaysia) Sdn. Bhd.
9th Floor
Chung Khiaw Bank Building
46, Jalan Raja Laut
KUALA LUMPUR
Tel: 03-986555
Telex: 31011 HPSM MA
A,C,E,M,P*

Protel Engineering
P.O.Box 1917
Lot 6624, Section 64
23/4 Pending Road
Kuching, **SARAWAK**
Tel: 36299
Telex: 70904 PROMAL MA
Cable: PROTELENG
A,E,M

MALTA
Philip Toledo Ltd.
Birkirkara P.O. Box 11
Notabile Rd.
MRIEHEL
Tel: 447 47, 455 66
Telex: 1649
E,M,P

MAURITIUS
Blanche Birger Co. Ltd.
18, Jules Koenig Street
PORT LOUIS
Tel: 20828
Telex: 4296
P

MEXICO
Hewlett-Packard de Mexico, S.A.
Francisco J. Allan #30
Colonia Nueva
Los Angeles 27140
COAHUILA, Torreon
Tel: 37220
P

Hewlett-Packard de Mexico, S.A.
Monti Morelos 299
Fraccionamiento Loma Bonita 45060
GUADALAJARA, Jalisco
Tel: 316630/314600
Telex: 0684 186 ECOME
P

MEXICO (Cont'd)

Microcomputadoras Hewlett-Packard, S.A.
Monti Pelvoux 115
LOS LOMAS, Mexico, D.F.
Tel: 520-9127
P

Hewlett-Packard Mexicana, S.A. de C.V.
Av. Periferico Sur No. 6501
Tepepan, Xochimilco
16020 **MEXICO D.F.**
Tel: 6-76-46-00
Telex: 17-74-507 HEWPACK MEX
A,C,CM,E,M,P

Hewlett-Packard De Mexico (Polanco)
Avenida Ejercito Nacional #579
$2^{da}y3^{er}$ piso
Colonia Granada 11560
MEXICO D.F.
Tel: 254-4433
P

Hewlett-Packard De Mexico, S.A. de C.V.
Czda. del Valle
409 Ote. 4th Piso
Colonia del Valle
Municipio de Garza Garciá
66220 **MONTERREY,** Nuevo León
Tel: 78 42 41
Telex: 038 410
P

MOROCCO

Etablissement Hubert Dolbeau & Fils
81 rue Karatchi
B.P. 11133
CASABLANCA
Tel: 3041-82, 3068-38
Telex: 23051, 22822
E

Gerep
2, rue Agadir
Boite Postale 156
CASABLANCA 01
Tel: 272093, 272095
Telex: 23 739
P

Sema-Maroc
Dept. Seric
6, rue Lapebie
CASABLANCA
Tel: 260980
Telex: 21641
C,P

NETHERLANDS

Hewlett-Packard Nederland B.V.
Startbaan 16
1187 XR **AMSTELVEEN**
P.O. Box 667
NL1180 AR **AMSTELVEEN**
Tel: (020) 547-6911
Telex: 13 216 HEPA NL
A,C,CM,E,M,P

Hewlett-Packard Nederland B.V.
Bongerd 2
NL 2906VK **CAPELLE A/D IJSSEL**
P.O. Box 41
NL 2900AA **CAPELLE A/D IJSSEL**
Tel: (10) 51-64-44
Telex: 21261 HEPAC NL
C,E

Hewlett-Packard Nederland B.V.
Pastoor Petersstraat 134-136
NL 5612 LV **EINDHOVEN**
P.O. Box 2342
NL 5600 CH **EINDHOVEN**
Tel: (040) 326911
Telex: 51484 hepae nl
A,C,E,M,P

NEW ZEALAND

Hewlett-Packard (N.Z.) Ltd.
5 Owens Road
P.O. Box 26-189
Epsom, **AUCKLAND**
Tel: 687-159
Cable: HEWPAK Auckland
C,CM,E,P*

Hewlett-Packard (N.Z.) Ltd.
4-12 Cruickshank Street
Kilbirnie, **WELLINGTON 3**
P.O. Box 9443
Courtenay Place, **WELLINGTON 3**
Tel: 877-199
Cable: HEWPACK Wellington
C,CM,E,P

Northrop Instruments & Systems Ltd.
369 Khyber Pass Road
P.O. Box 8602
AUCKLAND
Tel: 794-091
Telex: 60605
A,M

Northrop Instruments & Systems Ltd.
110 Mandeville St.
P.O. Box 8388
CHRISTCHURCH
Tel: 488-873
Telex: 4203
A,M

Northrop Instruments & Systems Ltd.
Sturdee House
85-87 Ghuznee Street
P.O. Box 2406
WELLINGTON
Tel: 850-091
Telex: NZ 3380
A,M

NIGERIA

Elmeco Nigeria Ltd.
46, Calcutta Crescent Apapa
P.O. Box 244and
LAGOS
E

NORTHERN IRELAND

See United Kingdom

NORWAY

Hewlett-Packard Norge A/S
Folke Bernadottes vei 50
P.O. Box 3558
N-5033 **FYLLINGSDALEN** (Bergen)
Tel: 0047/5/16 55 40
Telex: 76621 hpnas n
C,E,M

Hewlett-Packard Norge A/S
Osterndalen 16-18
P.O. Box 34
N-1345 **ÖSTERÅS**
Tel: 0047/2/17 11 80
Telex: 76621 hpnas n
A,C,CM,E,M,P

OMAN

Khimjil Ramdas
P.O. Box 19
MUSCAT/SULTANATE OF OMAN
Tel: 745601
Telex: 5289 BROKER MB MUSCAT
P

Suhail & Saud Bahwan
P.O.Box 169
MUSCAT/SULTANATE OF OMAN
Tel: 734201
Telex: 5274 BAHWAN MB
E

Imtac LLC
P.O. Box 8676
MUTRAH/SULTANATE OF OMAN
Tel: 601695
Telex: 5741 Tawoos On
A,C,M

PAKISTAN

Mushko & Company Ltd.
House No. 16, Street No. 16
Sector F-6/3
ISLAMABAD
Tel: 824545
Cable: FEMUS Islamabad
A,E,M,P*

Mushko & Company Ltd.
Oosman Chambers
Abdullah Haroon Road
KARACHI 0302
Tel: 524131, 524132
Telex: 2894 MUSKO PK
Cable: COOPERATOR Karachi
A,E,M,P*

PANAMA

Electronico Balboa, S.A.
Calle Samuel Lewis, Ed. Alfa
Apartado 4929
PANAMA 5
Tel: 64-2700
Telex: 3483 ELECTRON PG
A,CM,E,M,P

PERU

Cía Electro Médica S.A.
Los Flamencos 145, Ofc. 301/2
San Isidro
Casilla 1030
LIMA 1
Tel: 41-4325, 41-3705
Telex: Pub. Booth 25306 PEC PISIDR
CM,E,M,P

SAMS
Arenida Republica de Panama 3534
San Isidro, LIMA
Tel: 419928/417108
Telex: 20450 PE LIBERTAD
A,C,P

PHILIPPINES

The Online Advanced Systems Corp.
2nd Floor, Electra House
115-117 Esteban Street
Legaspi Village, Makati
P.O. Box 1510
Metro **MANILA**
Tel: 815-38-10 (up to 16)
Telex: 63274 ONLINE PN
A,C,E,M,P

PORTUGAL

Mundinter Intercambio
Mundial de Comércio S.A.R.L.
Av. Antonio Augusto Aguiar 138
Apartado 2761
LISBON
Tel: (19) 53-21-31, 53-21-37
Telex: 16691 munter p
M

Soquimica
Av. da Liberdade, 220-2
1298 **LISBOA** Codex
Tel: 56-21-82
Telex: 13316 SABASA
A

Telectra-Empresa Técnica de Equipmentos Eléctricos S.A.R.L.
Rua Rodrigo da Fonseca 103
P.O. Box 2531
LISBON 1
Tel: (19) 68-60-72
Telex: 12598
CM,E

C.P.C.S.I.
Rua de Costa Cabral 575
4200 **PORTO**
Tel: 499174/495173
Telex: 26054
C,P

PUERTO RICO

Hewlett-Packard Puerto Rico
101 Muñoz Rivera Av
Esu. Calle Ochoa
HATO REY, Puerto Rico 00918
Tel: (809) 754-7800
A,C,CM,M,E,P

QATAR

Computer Arabia
P.O. Box 2750
DOHA
Tel: 428555
Telex: 4806 CHPARB
P

Nasser Trading & Contracting
P.O.Box 1563
DOHA
Tel: 422170
Telex: 4439 NASSER DH
M

SAUDI ARABIA

Modern Electronics Establishment
Hewlett-Packard Division
P.O. Box 281
Thouqbah
AL-KHOBAR 31952
Tel: 895-1760, 895-1764
Telex: 671 106 HPMEEK SJ
Cable: ELECTA AL-KHOBAR
C,E,M

Modern Electronics Establishment
Hewlett-Packard Division
P.O. Box 1228
JEDDAH
Tel: 644 96 28
Telex: 4027 12 FARNAS SJ
Cable: ELECTA JEDDAH
A,C,CM,E,M,P

Modern Electronics Establishment
Hewlett-Packard Division
P.O.Box 22015
RIYADH 11495
Tel: 476-3030
Telex: 202049 MEERYD SJ
A,C,CM,E,M,P

Abdul Ghani El Ajou Corp.
P.O. Box 78
RIYADH
Tel: 40 41 717
Telex: 200 931 EL AJOU
P

SCOTLAND

See United Kingdom

SENEGAL

Societe Hussein Ayad & Cie.
76, Avenue Georges Pompidou
B.P. 305
DAKAR
Tel: 32339
Cable: AYAD-Dakar
E

Moneger Distribution S.A.
1, Rue Parent
B.P. 148
DAKAR
Tel: 215 671
Telex: 587
P

Systeme Service Conseil (SSC)
14, Avenue du Parachois
DAKAR ETOILE
Tel: 219976
Telex: 577
C,P

SINGAPORE

Hewlett-Packard Singapore (Sales) Pte. Ltd.
08-00 Inchcape House
450-2 Alexandra Road
Alexandra P.O. Box 58
SINGAPORE, 9115
Tel: 4731788
Telex: 34209 HPSGSO RS
Cable: HEWPACK, Singapore
A,C,E,M,P

Dynamar International Ltd.
Unit 05-11 Block 6
Kolam Ayer Industrial Estate
SINGAPORE 1334
Tel: 747-6188
Telex: 26283 RS
CM

SOUTH AFRICA

Hewlett-Packard So Africa (Pty.) Ltd.
P.O. Box 120
Howard Place **CAPE PROVINCE** 7450
Pine Park Center, Forest Drive, Pinelands
CAPE PROVINCE 7405
Tel: (021) 53 7954
Telex: 57-20006
A,C,CM,E,M,P

Hewlett-Packard So Africa (Pty.) Ltd.
2nd Floor Juniper House
92 Overport Drive
DURBAN 4067
Tel: (031) 28-4178
Telex: 6-22954
C

Hewlett-Packard So Africa (Pty.) Ltd.
6 Linton Arcade
511 Cape Road
Linton Grange
PORT ELIZABETH 6001
Tel: 041-301201
Telex: 24-2916
C

SALES & SUPPORT OFFICES
Arranged alphabetically by country

SOUTH AFRICA (Cont'd)
Hewlett-Packard So Africa (Pty.) Ltd.
Fountain Center
Kalkden Str.
Monument Park Ext 2
PRETORIA 0105
Tel: (012) 45 57258
Telex: 3-21063
C,E

Hewlett-Packard So Africa (Pty.) Ltd.
Private Bag Wendywood
SANDTON 2144
Tel: 802-5111, 802-5125
Telex: 4-20877 SA
Cable: HEWPACK Johannesburg
A,C,CM,E,M,P

SPAIN
Hewlett-Packard Española S.A.
Calle Entenza, 321
08029 **BARCELONA**
Tel: 3/322 24 51, 321 73 54
Telex: 52603 hpbee
A,C,E,M,P

Hewlett-Packard Española S.A.
Calle San Vicente S/N
Edificio Albia II-7B
48001 **BILBAO**
Tel: 4/423 83 06
A,C,E,M

Hewlett-Packard Española S.A.
Crta. de la Coruña, Km. 16, 400
Las Rozas
E-**MADRID**
Tel: (1) 637.00.11
Telex: 23515 HPE
C,M

Hewlett-Packard Española S.A.
Avda. S. Francisco Javier, S/N
Planta 10. Edificio Sevilla 2
41005 **SEVILLA**
Tel: 54/64 44 54
Telex: 72933
A,C,M,P

Hewlett-Packard Española S.A.
Isabel La Catolica, 8
46004 **VALENCIA**
Tel: 0034/6/351 59 44
C,P

SWEDEN
Hewlett-Packard Sverige AB
Ostra Tullgatan 3
S-21128 **MALMÖ**
Tel: (040) 70270
Telex: (854) 17886 (via Spånga office)
C,P

Hewlett-Packard Sverige AB
Skalholtsgatan 9, Kista
Box 19
S-16393 **SPÅNGA**
Tel: (08) 750-2000
Telex: (854) 17886
Telefax: (08) 7527781
A,C,CM,E,M,P

Hewlett-Packard Sverige AB
Frötallsgatan 30
S-42132 **VÄSTRA-FRÖLUNDA** (Gothenburg)
Tel: (031) 49-09-50
Telex: (854) 17886 (via Spånga office)
A,C,CM,E,M,P

SUDAN
Mediterranean Engineering & Trading Co. Ltd.
P.O. Box 1025
KHARTOUM
Tel: 41184
Telex: 24052
C,P

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Clarastrasse 12
CH-4058 **BASEL**
Tel: (61) 33-59-20
A

Hewlett-Packard (Schweiz) AG
7, rue du Bois-du-Lan
Case postale 365
CH-1217 **MEYRIN** 1
Tel: (0041) 22-83-11-11
Telex:27333 HPAG CH
C,CM

Hewlett-Packard (Schweiz) AG
Allmend 2
CH-8967 **WIDEN**
Tel: (0041) 57 31 21 11
Telex: 53933 hpag ch
Cable: HPAG CH
A,C,CM,E,M,P

SYRIA
General Electronic Inc.
Nuri Basha Ahnaf Ebn Kays Street
P.O. Box 5781
DAMASCUS
Tel: 33-24-87
Telex: 411 215
Cable: ELECTROBOR DAMASCUS
E

Middle East Electronics
P.O.Box 2308
Abu Rumaneh
DAMASCUS
Tel: 33 45 92
Telex: 411 771
M

TAIWAN
Hewlett-Packard Taiwan
Kaohsiung Office
11/F, 456, Chung Hsiao 1st Road
KAOHSIUNG
Tel: (07) 2412318
C,E

Hewlett-Packard Taiwan
8th Floor, Hewlett-Packard Building
337 Fu Hsing North Road
TAIPEI
Tel: (02) 712-0404
Telex: 24439 HEWPACK
Cable:HEWPACK Taipei
A,C,CM,E,M,P

Ing Lih Trading Co.
3rd Floor, 7 Jen-Ai Road, Sec. 2
TAIPEI 100
Tel: (02) 3948191
Cable: INGLIH Taipei
A

THAILAND
Unimesa Co. Ltd.
30 Patpong Ave., Suriwong
BANGKOK 5
Tel: 235-5727
Telex: 84439 Simonco TH
Cable: UNIMESA Bangkok
A,C,E,M

Bangkok Business Equipment Ltd.
5/5-6 Dejo Road
BANGKOK
Tel: 234-8670, 234-8671
Telex: 87699-BEQUIPT TH
Cable: BUSIQUIPT Bangkok
P

TOGO
Societe Africaine De Promotion
Immeuble Sagap
22, Rue d'Atakpame
B.P. 4150
LOME
Tel: 21-62-88
Telex: 5304
P

TRINIDAD & TOBAGO
Caribbean Telecoms Ltd.
Corner McAllister Street &
Eastern Main Road, Laventille
P.O. Box 732
PORT-OF-SPAIN
Tel: 624-4213
Telex: 22561 CARTEL WG
Cable: CARTEL, PORT OF SPAIN
CM,E,M,P

Computer and Controls Ltd.
P.O. Box 51
66 Independence Square
PORT-OF-SPAIN
Tel: 62-279-85
Telex: 3000 POSTLX WG, ACCT LOO9O AGENCY 1264
A,P

Feral Assoc.
8 Fitzgerald Lane
PORT-OF-SPAIN
Tel: 62-36864, 62-39255
Telex: 22432 FERALCO
Cable: FERALCO
M

TUNISIA
Precision Electronique S.A.R.L.
31 Avenue de la Liberte
TUNIS
Tel: 893937
Telex: 13238
P

Tunisie Electronique S.A.R.L.
94, Av. Jugurtha, Mutuelleville
1002 **TUNIS-BELVEDERE**
Tel: 280144
Telex: 13238
C,E,P

Corema S.A.
23, bis Rue de Marseille
TUNIS
Tel: 253-821
Telex: 14812 CABAM TN
M

TURKEY
E.M.A
Mediha Eldem Sokak No. 41/6
Yenisehir
ANKARA
Tel: 319175
Telex: 46912 KTX TR
Cable: EMATRADE ANKARA
M

Teknim Company Ltd.
Iran Caddesi No. 7
Kavaklidere
ANKARA
Tel: 275800
Telex: 42155 TKNM TR
E,CM

Saniva Bilgisayar Sistemleri A.S.
Buyukdere Caddesi 103/6
Gayrettene
ISTANBUL
Tel: 1727030
Telex: 26345 SANI TR
C,P

Best Inc.
Esentepe, Gazeteciler Sitesi
Keskin Kalemy
Sokak 6/3, Gayrettepe
ISTANBUL
Tel: 1721328
Telex: 42490
A

UNITED ARAB EMIRATES
Emitac Ltd.
P.O. Box 1641
SHARJAH
Tel: 591181
Telex: 68136 EMITAC EM
Cable: EMITAC SHARJAH
E,C,M,P,A

Emitac Ltd.
P.O. Box 2711
ABU DHABI
Tel: 820419-20
Cable: EMITACH ABUDHABI

Emitac Ltd.
P.O. Box 8391
DUBAI,
Tel: 377591

Emitac Ltd.
P.O. Box 473
RAS AL KHAIMAH
Tel: 28133, 21270

UNITED KINGDOM

GREAT BRITAIN
Hewlett-Packard Ltd.
Trafalgar House
Navigation Road
ALTRINCHAM
Cheshire WA14 1NU
Tel: 061 928 6422
Telex: 668068
A,C,E,M,P

Hewlett-Packard Ltd.
Miller House
The Ring, **BRACKNELL**
Berks RG12 1XN
Tel: 0344 424898
Telex: 848733
E

Hewlett-Packard Ltd.
Elstree House, Elstree Way
BOREHAMWOOD, Herts WD6 1SG
Tel: 01 207 5000
Telex: 8952716
C,E

Hewlett-Packard Ltd.
Oakfield House, Oakfield Grove
Clifton **BRISTOL,** Avon BS8 2BN
Tel: 0272 736806
Telex: 444302
C,E,P

Hewlett-Packard Ltd.
Bridewell House
9 Bridewell Place
LONDON EC4V 6BS
Tel: 01 583 6565
Telex: 298163
C,P

Hewlett-Packard Ltd.
Pontefract Road
NORMANTON, West Yorkshire WF6 1RN
Tel: 0924 895566
Telex: 557355
C,P

Hewlett-Packard Ltd.
The Quadrangle
106-118 Station Road
REDHILL, Surrey RH1 1PS
Tel: 0737 68655
Telex: 947234
C,E,P

Hewlett-Packard Ltd.
Avon House
435 Stratford Road
Shirley, **SOLIHULL,** West Midlands
B90 4BL
Tel: 021 745 8800
Telex: 339105
C,E,P

Hewlett-Packard Ltd.
West End House
41 High Street, West End
SOUTHAMPTON
Hampshire S03 3DQ
Tel: 0703 476767
Telex: 477138
C,P

Hewlett-Packard Ltd.
Harmon House
No. 1 George Street
UXBRIDGE, Middlesex UX8 1YH
Tel: 895 720 20
Telex: 893134/5
C,CM,E,M,P

Hewlett-Packard Ltd.
King Street Lane
Winnersh, **WOKINGHAM**
Berkshire RG11 5AR
Tel: 0734 784774
Telex: 847178
A,C,E,M,P

IRELAND

NORTHERN IRELAND
Hewlett-Packard (Ireland) Ltd.
Carrickfergus Industrial Centre
75 Belfast Road, Carrickfergus
BELFAST BT38 8PH
Tel: 09603 67333
Telex: 747626
C,E

SCOTLAND
Hewlett-Packard Ltd.
8 Woodside Place
GLASGOW, G3 7QF
Tel: 041 332 6232
Telex: 779615
C,E

Hewlett-Packard Ltd.
SOUTH QUEENSFERRY
West Lothian, EH30 9TG
Tel: 031 331 1188
Telex: 72682
C,CM,E,M,P

UNITED STATES

Alabama

Hewlett-Packard Co.
700 Century Park South, Suite 128
BIRMINGHAM, AL 35226
Tel: (205) 822-6802
A,C,M,P*

Hewlett-Packard Co.
420 Wynn Drive
HUNTSVILLE, AL 35805
Tel: (205) 830-2000
C,CM,E,M*

Alaska

Hewlett-Packard Co.
3601 C St., Suite 1416
ANCHORAGE, AK 99503
Tel: (907) 563-8855
C,E

Arizona

Hewlett-Packard Co.
8080 Pointe Parkway West
PHOENIX, AZ 85044
Tel: (602) 273-8000
A,C,CM,E,M,P

Hewlett-Packard Co.
3400 East Britannia Dr.
Bldg. C, Suite 124
TUCSON, AZ 85706
Tel: (602) 573-7400
C,E,M**

California

Hewlett-Packard Co.
99 South Hill Dr.
BRISBANE, CA 94005
Tel: (415) 330-2500
C

Hewlett-Packard Co.
5060 E. Clinton Avenue, Suite 102
FRESNO, CA 93727
Tel: (209) 252-9652
C,M

Hewlett-Packard Co.
1421 S. Manhattan Av.
FULLERTON, CA 92631
Tel: (714) 999-6700
C,CM,E,M

Hewlett-Packard Co.
7408 Hollister Ave. #A
GOLETA, CA 93117
Tel: (805) 685-6100
C,E

Hewlett-Packard Co.
5400 W. Rosecrans Blvd.
LAWNDALE, CA 90260
Tel: (213) 643-7500
Telex: 910-325-6608
C,M

Hewlett-Packard Co.
2525 Grand Avenue
Long Beach, CA 90815
Tel: (213) 498-1111
C

Hewlett-Packard Co.
3155 Porter Drive
PALO ALTO, CA 94304
Tel: (415) 857-8000
C,E

Hewlett-Packard Co.
4244 So. Market Court, Suite A
SACRAMENTO, CA 95834
Tel: (916) 929-7222
A*,C,E,M

Hewlett-Packard Co.
9606 Aero Drive
SAN DIEGO, CA 92123
Tel: (619) 279-3200
C,CM,E,M

Hewlett-Packard Co.
5725 W. Las Positas Blvd.
Pleasanton, CA 94566
Tel: (415) 460-0282
C

Hewlett-Packard Co.
3003 Scott Boulevard
SANTA CLARA, CA 95054
Tel: (408) 988-7000
Telex: 910-338-0586
A,C,CM,E

Hewlett-Packard Co.
2150 W. Hillcrest Dr.
THOUSAND OAKS, CA 91320
(805) 373-7000
C,CM,E

Colorado

Hewlett-Packard Co.
2945 Center Green Court South
Suite A
BOULDER, CO 80301
Tel: (303) 938-3005
A,C,E

Hewlett-Packard Co.
24 Inverness Place, East
ENGLEWOOD, CO 80112
Tel: (303) 649-5000
A,C,CM,E,M

Connecticut

Hewlett-Packard Co.
500 Sylvan Av.
BRIDGEPORT, CT 06606
Tel: (203) 371-6454
C,E

Hewlett-Packard Co.
47 Barnes Industrial Road South
WALLINGFORD, CT 06492
Tel: (203) 265-7801
A,C,CM,E,M

Florida

Hewlett-Packard Co.
2901 N.W. 62nd Street
FORT LAUDERDALE, FL 33309
Tel: (305) 973-2600
C,E,M,P*

Hewlett-Packard Co.
6800 South Point Parkway
Suite 301
JACKSONVILLE, FL 32216
Tel: (904) 398-0663
C*,M**

Hewlett-Packard Co.
6177 Lake Ellenor Drive
ORLANDO, FL 32809
Tel: (305) 859-2900
A,C,CM,E,P*

Hewlett-Packard Co.
4700 Bayou Blvd.
Building 5
PENSACOLA, FL 32503
Tel: (904) 476-8422
A,C,M

Hewlett-Packard Co.
5550 W. Idlewild, 150
TAMPA, FL 33614
Tel: (813) 884-3282
C,E,M,P

Georgia

Hewlett-Packard Co.
2000 South Park Place
ATLANTA, GA 30339
Tel: (404) 955-1500
Telex: 810-766-4890
A,C,CM,E,M,P*

Hewlett-Packard Co.
3607 Parkway Lane
Suite 300
NORCROSS, GA 30092
Tel: (404) 448-1894
C,E,P

Hawaii

Hewlett-Packard Co.
Kawaiahao Plaza, Suite 190
567 South King Street
HONOLULU, HI 96813
Tel: (808) 526-1555
A,C,E,M

Idaho

Hewlett-Packard Co.
11309 Chinden Blvd.
BOISE, ID 83707
Tel: (208) 323-2700
C

Illinois

Hewlett-Packard Co.
304 Eldorado Road
P.O. Box 1607
BLOOMINGTON, IL 61701
Tel: (309) 662-9411
C,M**

Hewlett-Packard Co.
525 W. Monroe, 1308
CHICAGO, IL 60606
Tel: (312) 930-0010
C

Hewlett-Packard Co.
1200 East Diehl Road
NAPERVILLE, IL 60566
Tel: (312) 357-8800
C

Hewlett-Packard Co.
5201 Tollview Drive
ROLLING MEADOWS, IL 60008
Tel: (312) 255-9800
Telex: 910-687-1066
A,C,CM,E,M

Indiana

Hewlett-Packard Co.
11911 N. Meridian St.
CARMEL, IN 46032
Tel: (317) 844-4100
A,C,CM,E,M

Hewlett-Packard Co.
3702 Rupp Drive
FT. WAYNE, IN 46815
Tel: (219) 482-4283
C,E

Iowa

Hewlett-Packard Co.
4070 22nd Av. SW
CEDAR RAPIDS, IA 52404
Tel: (319) 390-4250
C,E,M

Hewlett-Packard Co.
4201 Corporate Dr.
WEST DES MOINES, IA 50265
Tel: (515) 224-1435
A**,C,M**

Kansas

Hewlett-Packard Co.
7804 East Funston Road, 203
WICHITA, KS 67207
Tel: (316) 684-8491
C,E

Kentucky

Hewlett-Packard Co.
10300 Linn Station Road, 100
LOUISVILLE, KY 40223
Tel: (502) 426-0100
A,C,M

Louisiana

Hewlett-Packard Co.
160 James Drive East
ST. ROSE, LA 70087
P.O. Box 1449
KENNER, LA 70063
Tel: (504) 467-4100
A,C,E,M,P

Maryland

Hewlett-Packard Co.
3701 Koppers Street
BALTIMORE, MD 21227
Tel: (301) 644-5800
Telex: 710-862-1943
A,C,CM,E,M

Hewlett-Packard Co.
2 Choke Cherry Road
ROCKVILLE, MD 20850
Tel: (301) 948-6370
A,C,CM,E,M

Massachusetts

Hewlett-Packard Co.
1775 Minuteman Road
ANDOVER, MA 01810
Tel: (617) 682-1500
A,C,CM,E,M,P*

Hewlett-Packard Co.
32 Hartwell Avenue
LEXINGTON, MA 02173
Tel: (617) 861-8960
C,E

Michigan

Hewlett-Packard Co.
4326 Cascade Road S.E.
GRAND RAPIDS, MI 49506
Tel: (616) 957-1970
C,M

Hewlett-Packard Co.
39550 Orchard Hill Place Drive
NOVI, MI 48020
Tel: (313) 349-9200
A,C,E,M

Hewlett-Packard Co.
1771 W. Big Beaver Road
TROY, MI 48084
Tel: (313) 643-6474
C

Minnesota

Hewlett-Packard Co.
2025 W. Larpenteur Ave.
ST. PAUL, MN 55113
Tel: (612) 644-1100
A,C,CM,E,M

Missouri

Hewlett-Packard Co.
1001 E. 101st Terrace Suite 120
KANSAS CITY, MO 64131-3368
Tel: (816) 941-0411
A,C,CM,E,M

Hewlett-Packard Co.
13001 Hollenberg Drive
BRIDGETON, MO 63044
Tel: (314) 344-5100
A,C,E,M

Nebraska

Hewlett-Packard
10824 Old Mill Rd., Suite 3
OMAHA, NE 68154
Tel: (402) 334-1813
C,E,M

New Jersey

Hewlett-Packard Co.
120 W. Century Road
PARAMUS, NJ 07653
Tel: (201) 265-5000
A,C,CM,E,M

Hewlett-Packard Co.
20 New England Av. West
PISCATAWAY, NJ 08854
Tel: (201) 562-6100
A,C,CM,E

New Mexico

Hewlett-Packard Co.
7801 Jefferson N.E.
ALBUQUERQUE, NM 87109
Tel: (505) 292-1330
C,E,M

New York

Hewlett-Packard Co.
5 Computer Drive South
ALBANY, NY 12205
Tel: (518) 458-1550
A,C,E,M

Hewlett-Packard Co.
9600 Main Street
CLARENCE, NY 14031
Tel: (716) 759-8621
C,E

Hewlett-Packard Co.
200 Cross Keys Office Park
FAIRPORT, NY 14450
Tel: (716) 223-9950
A,C,CM,E,M

Hewlett-Packard Co.
7641 Henry Clay Blvd.
LIVERPOOL, NY 13088
Tel: (315) 451-1820
A,C,CM,E,M

Hewlett-Packard Co.
No. 1 Pennsylvania Plaza
55th Floor
34th Street & 8th Avenue
MANHATTAN NY 10119
Tel: (212) 971-0800
C,M*

Hewlett-Packard Co.
15 Myers Corner Rd.
Hollowbrook Park, Suite 20
WAPPINGER FALLS, NY 12590
CM,E

Hewlett-Packard Co.
250 Westchester Avenue
WHITE PLAINS, NY 10604
Tel: (914) 684-6100
C,CM,E

Hewlett-Packard Co.
3 Crossways Park West
WOODBURY, NY 11797
Tel: (516) 682-7800
A,C,CM,E,M

SALES & SUPPORT OFFICES

Arranged alphabetically by country

UNITED STATES (Cont'd)

North Carolina
Hewlett-Packard Co.
305 Gregson Dr.
CARY, NC 27511
Tel: (919) 467-6600
C,CM,E,M,P*

Hewlett-Packard Co.
9600-H Southern Pine Blvd.
CHARLOTTE, NC 28210
Tel: (704) 527-8780
C*

Hewlett-Packard Co.
5605 Roanne Way
GREENSBORO, NC 27420
Tel: (919) 852-1800
A,C,CM,E,M,P*

Ohio
Hewlett-Packard Co.
2717 S. Arlington Road
AKRON, OH 44312
Tel: (216) 644-2270
C,E

Hewlett-Packard Co.
23200 Chagrin Blvd #100
BEACHWOOD, OH 44122
Tel: (216) 292-4677
C,P

Hewlett-Packard Co.
9920 Carver Road
CINCINNATI, OH 45242
Tel: (513) 891-9870
C,M

Hewlett-Packard Co.
16500 Sprague Road
CLEVELAND, OH 44130
Tel: (216) 243-7300
A,C,CM,E,M

Hewlett-Packard Co.
9080 Springboro Pike
MIAMISBURG, OH 45342
Tel: (513) 433-2223
A,C,CM,E*,M

Hewlett-Packard Co.
One Maritime Plaza, 5th Floor
720 Water Street
TOLEDO, OH 43604
Tel: (419) 242-2200
C

Hewlett-Packard Co.
675 Brooksedge Blvd.
WESTERVILLE, OH 43081
Tel: (614) 891-3344
C,CM,E*

Oklahoma
Hewlett-Packard Co.
3525 N.W. 56th St.
Suite C-100
OKLAHOMA CITY, OK 73112
Tel: (405) 946-9499
C,E*,M

Hewlett-Packard Co.
3840 S. 103rd E. Ave., 100
TULSA, OK 74146
Tel: (918) 665-3300
A**,C,E,M*,P*

Oregon
Hewlett-Packard Co.
9255 S. W. Pioneer Court
WILSONVILLE, OR 97070
Tel: (503) 682-8000
A,C,E*,M

Pennsylvania
Hewlett-Packard Co.
50 Dorchester Rd.
HARRISBURG, PA 17112
Tel: (717) 657-5900
C

Hewlett-Packard Co.
111 Zeta Drive
PITTSBURGH, PA 15238
Tel: (412) 782-0400
A,C,E,M

Hewlett-Packard Co.
2750 Monroe Boulevard
VALLEY FORGE, PA 19482
Tel: (215) 666-9000
A,C,CM,E,M

South Carolina
Hewlett-Packard Co.
Brookside Park, Suite 122
1 Harbison Way
COLUMBIA, SC 29210
Tel: (803) 732-0400
C,M

Hewlett-Packard Co.
555 N. Pleasantburg Dr.
Suite 107
GREENVILLE, SC 29607
Tel: (803) 232-8002
C

Tennessee
Hewlett-Packard Co.
One Energy Centr. 200
Pellissippi Pkwy.
KNOXVILLE, TN 37932
Tel: (615) 966-4747
A,C,M

Hewlett-Packard Co.
3070 Directors Row
Directors Square
MEMPHIS, TN 38131
Tel: (901) 346-8370
A,C,M

Hewlett-Packard Co.
220 Great Circle Road, Suite 116
NASHVILLE, TN 37228
Tel: (615) 255-1271
C,M,P*

Texas
Hewlett-Packard Co.
1826-P Kramer Lane
AUSTIN, TX 78758
Tel: (512) 835-6771
C,E,P*

Hewlett-Packard Co.
5700 Cromo Dr
EL PASO, TX 79912
Tel: (915) 833-4400
C,E*,M**

Hewlett-Packard Co.
3952 Sandshell Drive
FORT WORTH, TX 76137
Tel: (817) 232-9500
C

Hewlett-Packard Co.
10535 Harwin Drive
HOUSTON, TX 77036
Tel: (713) 776-6400
A,C,E,M,P*

Hewlett-Packard Co.
511 E. John W. Carpenter Fwy.
Royal Tech. Center 100
IRVING, TX 75062
Tel: (214) 556-1950
C,E

Hewlett-Packard Co.
109 E. Toronto, Suite 100
McALLEN, TX 78503
Tel: (512) 630-3030
C

Hewlett-Packard Co.
930 E. Campbell Rd.
RICHARDSON, TX 75081
Tel: (214) 231-6101
A,C,CM,E,M,P*

Hewlett-Packard Co.
1020 Central Parkway South
SAN ANTONIO, TX 78216
Tel: (512) 494-9336
A,C,E,M,P*

Utah
Hewlett-Packard Co.
3530 W. 2100 South
SALT LAKE CITY, UT 84119
Tel: (801) 974-1700
A,C,E,M

Virginia
Hewlett-Packard Co.
4305 Cox Road
GLEN ALLEN, VA 23060
Tel: (804) 747-7750
A,C,E,M,P*

Hewlett-Packard Co.
Tanglewood West Bldg.
Suite 240
3959 Electric Road
ROANOKE, VA 24018
Tel: (703) 774-3444
C,E,P

Washington
Hewlett-Packard Co.
15815 S.E. 37th Street
BELLEVUE, WA 98006
Tel: (206) 643-4000
A,C,CM,E,M

Hewlett-Packard Co.
708 North Argonne Road
SPOKANE, WA 99212-2793
Tel: (509) 922-7000
C

West Virginia
Hewlett-Packard Co.
501 56th
CHARLESTON, WV 25304
Tel: (304) 925-0492
A,C,M

Wisconsin
Hewlett-Packard Co.
275 N. Corporate Dr.
BROOKFIELD, WI 53005
Tel: (414) 784-8800
A,C,E*,M

URUGUAY

Pablo Ferrando S.A.C. e I.
Avenida Italia 2877
Casilla de Correo 370
MONTEVIDEO
Tel: 80-2586
Telex: 802586
A,CM,E,M

Olympia de Uruguay S.A.
Maquines de Oficina
Avda. del Libertador 1997
Casilla de Correos 6644
MONTEVIDEO
Tel: 91-1809, 98-3807
Telex: 6342 OROU UY
P

VENEZUELA

Hewlett-Packard de Venezuela C.A.
3A Transversal Los Ruices Norte
Edificio Segre 2 & 3
Apartado 50933
CARACAS 1071
Tel: 239-4133
Telex: 251046 HEWPACK
A,C,CM,E,M,P

Hewlett-Packard de Venezuela, C.A.
Centro Civdad Comercial Tamanaco
Nivel C-2 (Nueva Etapa)
Local 53H05
Chuao, **CARACAS**
Tel: 928291
P

Albis Venezolana S.R.L.
Av. Las Marias, Ota. Alix,
El Pedregal
Apartado 81025
CARACAS 1080A
Tel: 747984, 742146
Telex: 24009 ALBIS VC
A

Tecnologica Medica del Caribe, C.A.
Multicentro Empresarial del Este
Ave. Libertador
Edif. Libertador
Nucleo "C" - Oficina 51-52
CARACAS
Tel: 339867/333780
M

Hewlett-Packard de Venezuela C.A.
Residencias Tia Betty Local 1
Avenida 3 y con calfe 75
MARACAIBO, Estado Zulia
Apartado 2646
Tel: (061) 75801-75805-75806-80304
Telex: 62464 HPMAR
C,E*

Hewlett-Packard de Venezuela C.A.
Urb. Lomas de Este
Torre Trebol — Piso 11
VALENCIA, Estado Carabobo
Apartado 3347
Tel: (041) 222992/223024
C,P

YUGOSLAVIA

Do Hermes
General Zdanova 4
YU-11000 **BEOGRAD**
Tel: 340 327, 342 641
Telex: 11433
A,C,E,P

Hermes
Titova 50
YU-61000 **LJUBLJANA**
Tel: 324 856, 324 858
Telex: 31583
C,E,M,P

Elektrotehna
Titova 51
YU-61000 **LJUBLJANA**
CM

ZAIRE

Computer & Industrial Engineering
25, Avenue de la Justice
B.P. 12797
KINSHASA, Gombe
Tel: 32063
Telex: 21552
C,P

ZAMBIA

R.J. Tilbury (Zambia) Ltd.
P.O. Box 32792
LUSAKA
Tel: 215590
Telex: 40128
E

ZIMBABWE

Field Technical Sales (Private) Limited
45, Kelvin Road North
P.O. Box 3458
HARARE
Tel: 705 231
Telex: 4-122 RH
E,P

Manual Part Number 64341-90903
Printed in U.S.A., SEPTEMBER 1985
E0985
Replaces 64341-90901, February 1985

HEWLETT PACKARD